	L #	Hits	Search Text	DBs
1	L1	6228	(operation instruction command) near20 ((issu$3 dispatch$3 schedul$3 register) near10 (port slot))	USPAT; US-PGPUB
2	L2	246	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 1	USPAT; US-PGPUB
3	L3	556	(long compound) adj2 instruction and 1	USPAT; US-PGPUB
4	L4	154275	(issu$3 dispatch$3 schedul$3 (port slot)).ab,ti.	USPAT; US-PGPUB
5	L6	185	3 and 4	USPAT; US-PGPUB
6	L7	15	2 and 6	USPAT; US-PGPUB

FIG. 59

	VALID	ADDRESS	DISP	INSTRUCTION
#0				
#1				
:	:	:	:	:
:	:	:	:	:
#n				

FIG. 60

	VALID	ADDRESS	MODE	INSTRUCTION
#0				
#1				
:	:	:	:	:
:	:	:	:	:
#n				

FIG. 61

	VALID	ADDRESS	DISP	MODE	INSTRUCTION
#0					
#1					
:	:	:	:	:	:
:	:	:	:	:	:
#n					

	Document ID	U	Title	Current OR
1	US 20040010635 A1	☐	An architecture and related methods facilitating secure port bypass circuit settings	710/1
2	US 20040007562 A1	☒	Welding parameters setting method for a resistance welder controll apparatus	219/86.41
3	US 20040003212 A1	☒	Data processor	712/229
4	US 20030236953 A1	☒	System and method for providing multi-initiator capability to an ATA drive	711/151
5	US 20030236952 A1	☒	System and method for providing multi-initiator capability to an ATA drive	711/151
6	US 20030212841 A1	☒	Method and apparatus of controlling an operational mode of a USB device	710/62
7	US 20030202517 A1	☒	Apparatus for controlling packet output	370/395.4
8	US 20030153988 A1	☒	Highly versatile process control system controller	700/19
9	US 20030126233 A1	☒	Content service aggregation system	709/219
10	US 20030110344 A1	☒	Communications systems, apparatus and methods	711/100
11	US 20030109307 A1	☒	Method and apparatus for communicating with a player of a networked gaming device	463/41
12	US 20030088479 A1	☒	Online scheduling system	705/26
13	US 20030043836 A1	☒	Cross-bar switch with sink port accepting multiple packets	370/424
14	US 20030043835 A1	☒	Cross-bar switch with explicit multicast support	370/424
15	US 20030043829 A1	☒	Cross-bar switch employing a multiple entry point FIFO	370/412
16	US 20030043818 A1	☒	Cross-bar switch with bandwidth allocation	370/395.42
17	US 20030043817 A1	☒	Cross-bar switch supporting implicit multicast addressing	370/395.31

unlike the 11th embodiment in which the determination is done by the hardware mechanism in the VLIW type processor, a flow chart shown in FIG. **49** is used. Referring to FIG. **49**, in step **S15**, it is determined whether the long instruction word as a breakpoint target includes a short instruction formed from a conditional instruction. On the basis of the determination result, it is determined in step **S16** whether the long instruction word as a breakpoint target contains a conditional instruction.

When short instructions forming the long instruction word contain only unconditional instructions, the flow jumps to step **S10**. When a conditional instruction is contained, the flow advances to step **S17**. It is determined in step **S17** by referring to the condition register **51** whether the condition of the conditional instruction contained in the long instruction word as a breakpoint target is satisfied. In step **S8**, it is determined whether the condition of the conditional instruction is satisfied. Processing in the remaining, steps is the same as described above.

In this case, when the short instructions forming the long instruction word include a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the conditional instruction is satisfied for any one of the short instructions, a break-interrupt occurs. When the condition of the conditional instruction is not satisfied for none of the short instructions, or all the short instructions are unconditional instructions, a break-interrupt can be inhibited.

To implement determination of the condition of a conditional instruction by the function of software, unlike the 12th embodiment in which the determination is done by the hardware mechanism of the VLIW type processor, a flow chart shown in FIG. **50** is used. Referring to FIG. **50**, in step **S25**, it is determined whether a short instruction of interest in short instructions forming a long instruction word as a breakpoint target is a conditional instruction. In step **S26**, it is determined whether the short instruction of interest is a conditional instruction.

The short instruction in the long instruction word is specified using the breakpoint table held in the instruction execution section **30**. More specifically, the breakpoint table has the construction shown in FIG. **51**. A column "DISP" is added to the breakpoint table shown in FIG. **48**. The column "DISP" holds displacement information from the head portion of the long instruction word as a breakpoint target. One of the short instructions forming the long instruction word is specified using the displacement information together with the instruction break address held in the column "ADDRESS".

If it is determined in step **S26** that the short instruction of interest is not a conditional instruction, the flow jumps to step **S10**. If the short instruction is a conditional instruction, the flow advances to step **S27**. It is determined in step **S27** by referring to the condition register **51** whether the condition of the conditional instruction of interest in the long instruction word as a breakpoint target is satisfied. In step **S8**, it is determined whether the condition of the conditional instruction is satisfied. Processing in the remaining steps is the same as described above.

In this case, one of the short instructions forming the long instruction word is specified, and it is determined whether the condition of the conditional instruction is satisfied. A break-interrupt can be controlled in accordance with whether the condition is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the designated conditional instruction is satisfied, a break-interrupt occurs. When the condition of the designated conditional instruction is not satisfied, or the designated short instruction is an unconditional instruction, a break-interrupt can be inhibited.

To implement determination of the condition of a conditional instruction by the function of software, unlike the 13th embodiment in which the determination is done by the hardware mechanism of the parallel processor, a flow chart shown in FIG. **52** is used. Referring to FIG. **52**, in step **S35**, it is determined whether a variable-length instruction word as a breakpoint target contains a basic instruction. On the basis of this determination result, it is determined in step **S36** whether the variable-length instruction word contains a conditional instruction.

When basic instructions forming the breakpoint target instruction word contain only unconditional instructions, the flow jumps to step **S10**. When a conditional instruction is contained, the flow advances to step **S37**. It is determined in step **S37** by referring to the condition register **51** whether the condition of the conditional instruction contained in the breakpoint target instruction word is satisfied. In step **S8**, it is determined whether the condition of the conditional instruction is satisfied. Processing in the remaining steps is the same as described above.

In this case, when the basic instructions forming the variable-length instruction word include a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the conditional instruction is satisfied for any one of the basic instructions, a break-interrupt occurs. When the condition of the conditional instruction is not satisfied for none of the basic instructions, or all the basic instructions are unconditional instructions, a break-interrupt can be inhibited.

To implement determination of the condition of a conditional instruction by the function of software, unlike the 14th embodiment in which the determination is done by the hardware mechanism of the parallel processor, a flow chart shown in FIG. **53** is used. Referring to FIG. **53**, in step **S45**, it is determined whether a basic instruction of interest in basic instructions forming a variable-length instruction word as a breakpoint target is a conditional instruction. In step **S46**, it is determined whether the basic instruction of interest is a conditional instruction.

The basic instruction in the variable-length instruction word is specified using a breakpoint table shown in FIG. **51**. In this breakpoint table, a column "DISP" holds displacement information from the head portion of the variable-length instruction word as a breakpoint target. One of the basic instructions forming the variable-length instruction word is specified using the displacement information together with the instruction break address held in the column "ADDRESS".

If it is determined in step **S46** that the basic instruction of interest is not a conditional instruction, the flow jumps to step **S10**. If the basic instruction is a conditional instruction, the flow advances to step **S47**. It is determined in step **S47** by referring to the condition register **51** whether the condition of the conditional instruction of interest in the breakpoint target instruction word is satisfied. In step **S8**, it is determined whether the condition of the conditional instruction is satisfied. Processing in the remaining steps is the same as described above.

	Document ID	U	Title	Current OR
18	US 20030043812 A1	☒	Cross-bar switch incorporating a sink port with retry capability	370/395.4
19	US 20030043797 A1	☒	Cross-bar switch	370/389
20	US 20030043771 A1	☒	CONNECTION ESTABLISHMENT METHOD, COMMUNICATION METHOD, STATE CHANGE TRANSMISSION METHOD, STATE CHANGING METHOD WIRELESS APPARATUS, WIRELESS DEVICE, AND COMPUTER	370/338
21	US 20030021230 A1	☒	Switch fabric with bandwidth efficient flow control	370/230
22	US 20030002537 A1	☒	Method and apparatus for controlling the timing of a communication device	370/503
23	US 20020169521 A1	☒	AUTOMATED DATA STORAGE LIBRARY WITH MULTIPURPOSE SLOTS PROVIDING USER-SELECTED CONTROL PATH TO SHARED ROBOTIC DEVICE	700/245
24	US 20020159385 A1	☒	Link level packet flow control mechanism	370/229
25	US 20020144048 A1	☒	Data storage media library with scalable throughput rate for data routing and protocol conversion	711/4
26	US 20020138834 A1	☒	System and method for displaying advertising in an interactive program guide	725/42
27	US 20020112142 A1	☒	IMPLEMENTATION OF A CONDITIONAL MOVE INSTRUCTION IN AN OUT-OF-ORDER PROCESSOR	712/8
28	US 20020083304 A1	☒	Rename finish conflict detection and recovery	712/218
29	US 20010027538 A1	☒	Computer register watch	714/28
30	US 20010016901 A1	☒	Communicating instruction results in processors and compiling methods for processors	712/217
31	US 6658002 B1	☒	Logical operation unit for packet processing	370/392
32	US 6651247 B1	☒	Method, apparatus, and product for optimizing compiler with rotating register assignment to modulo scheduled code in SSA form	717/161
33	US 6647433 B1	☒	Architecture and related methods facilitating secure port bypass circuit settings	710/5
34	US 6603744 B2	☒	Connection establishment method, communication method, state change transmission method, state changing method, wireless apparatus, wireless device, and computer	370/310
35	US 6591322 B1	☒	Method and apparatus for connecting single master devices to a multimaster wired-and bus environment	710/110
36	US 6522674 B1	☒	Use of pointers to enhance flexibility of serial port interface for an integrated circuit with programmable components	341/141

In this case, one of the basic instructions forming the variable-length instruction word is specified, and it is determined whether the condition of the conditional instruction is satisfied. A break-interrupt can be controlled in accordance with whether the condition is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the designated conditional instruction is satisfied, a break-interrupt occurs. When the condition of the designated conditional instruction is not satisfied, or the designated basic instruction is an unconditional instruction, a break-interrupt can be inhibited.

19th Embodiment

The 19th embodiment of the present invention will be described next with reference to drawings.

FIG. **54** is a flow chart showing the processing procedure of an interrupt handler according to the 19th embodiment. In FIG. **54**, the same step numbers as in FIG. **47** denote the same processing contents as in FIG. **47**.

In the 19th embodiment, interrupt control based on determination of the condition of a conditional instruction, which is done by the hardware mechanism in the processor of the 15th embodiment, is performed by the function of software shown in the flow chart of FIG. **54**. More specifically, in this embodiment, a function of switching between an instruction break mode in which an instruction break occurs when the instruction break generation condition is satisfied and a conditional instruction break mode in which an instruction break occurs when both the instruction break generation condition and the condition of the conditional instruction are satisfied is used.

The breakpoint table according to the 19th embodiment has the construction shown in FIG. **55**. A column "MODE" is added to the breakpoint table shown in FIG. **48**. The column "MODE" holds mode information on instruction break mode or conditional instruction break mode. In the mode information, the value "0" means the instruction break mode, while the value "1" means the conditional instruction break mode.

Referring to FIG. **54**, in this embodiment, when it is determined in step S3 that an entry corresponding to the break-interrupt generation address is found in the breakpoint table shown in FIG. **55**, the flow advances to step S12. In step S12, it is determined by looking up the breakpoint table shown in FIG. **55** whether the instruction break mode or the conditional instruction break mode is set for the entry.

If the instruction break mode is set, the flow immediately advances to step S9 to execute processing for the instruction break because this mode generates a break-interrupt when the instruction break generation condition is satisfied. If the conditional instruction break mode is set, the flow advances to step S5 because this mode generates a break-interrupt only when the condition of the conditional instruction is also satisfied. Processing in the remaining steps is the same as described above with reference to FIG. **47**.

As described above, in the 19th embodiment, whether the instruction break generation condition related to the instruction break address and the flag value is satisfied is determined by each of determination sections 25_0 to 25_n. In the instruction break mode, when the instruction break generation condition is satisfied, a break-interrupt occurs. In the conditional instruction break mode, it is also determined by software of the interrupt handler whether the condition of the conditional instruction is satisfied. Only when both conditions are satisfied, a break-interrupt actually occurs.

Thus, like in the 15th embodiment, when the conditional instruction break,mode is designated, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. In addition, when the instruction break mode is designated, a break-interrupt can be controlled independently of whether the condition of the conditional instruction is satisfied and in accordance with whether the instruction break generation condition is satisfied.

In the example shown in FIG. **54**, processing corresponding to each determination section of the scalar processing shown in FIG. **29** is implemented by the function of software. Processing corresponding to each determination section of the VLIW type processor described in each of the embodiments shown in FIGS. **30** and **31** or processing corresponding to each determination section of the parallel processor described in each of the embodiments shown in FIGS. **32** and **33** can also be implemented by the function of software.

For example, to apply the function to the VLIW type processor, processing in steps S5 to S7 in the flow chart of FIG. **54** is replaced by processing in steps S15 to S17 shown in FIG. **49** or processing in steps S25 to S27 shown in FIG. **50**. To apply the function to the parallel processor, processing in steps S5 to S7 in the flow chart of FIG. **54** is replaced by processing in steps S35 to S37 shown in FIG. **52** or processing in step's S45 to S47 shown in FIG. **53**.

However, when the processing is replaced by processing in steps S25 to S27 shown in FIG. **50** or in steps S45 to S47 in FIG. **53**, a breakpoint table shown in FIG. **56** must be used.

20th Embodiment

The 20th embodiment of the present invention will be described next with reference to drawings.

FIG. **57** is a flow chart showing the processing procedure of an interrupt handler according to the 20th embodiment. In FIG. **57**, the same step numbers as in FIG. **47** denote the same processing contents as in FIG. **47**. The breakpoint table has the same construction as in FIG. **48**.

In the 20th embodiment, interrupt control based on determination of the condition of a conditional instruction, which is done by the hardware mechanism in the processor of the 16th embodiment, is performed by the function of software shown in the flow chart of FIG. **57**. More specifically in this embodiment, a function of generating a break-interrupt not only when the condition of a conditional instruction is satisfied but also when the breakpoint target instruction word is an unconditional instruction.

In the 18th embodiment shown in FIG. **47**, if it is determined in step S6 that the breakpoint target instruction word is not a conditional instruction, the flow immediately jumps to step S10 without performing processing for the instruction break in step S9. However, in the 20th embodiment shown in FIG. **57**, even when it is determined in step S6 that the breakpoint target instruction word is not a conditional instruction, the flow advances to step S9 to execute processing for the instruction break.

As described above, in the 20th embodiment, whether the instruction break generation condition related to the instruction break address and the flag value is satisfied is determined by each of determination sections 25_0 to 25_n. When the condition is satisfied, it is also determined whether the breakpoint target instruction word is a conditional instruction. If the breakpoint target instruction word is an unconditional instruction, a break-interrupt unconditionally occurs. If the breakpoint target instruction word is a condi-

	Document ID	U	Title	Current OR
37	US 6489933 B1	☒	Display controller with motion picture display function, computer system, and motion picture display control method	345/1.1
38	US 6487474 B1	☒	Automated data storage library with multipurpose slots providing user-selected control path to shared robotic device	700/245
39	US 6480097 B1	☒	Security control for personal computer	340/5.8
40	US 6460129 B1	☒	Pipeline operation method and pipeline operation device to interlock the translation of instructions based on the operation of a non-pipeline operation unit	712/31
41	US 6449713 B1	☒	Implementation of a conditional move instruction in an out-of-order processor	712/234
42	US 6425100 B1	☒	Snoopy test access port architecture for electronic circuits including embedded core with built-in test access port	714/724
43	US 6404707 B1	☒	Storage apparatus using removable media and its read/write control method	369/30.06
44	US 6381717 B1	☒	Snoopy test access port architecture for electronic circuits including embedded core having test access port with instruction driven wake-up	714/724
45	US 6378090 B1	☒	Hierarchical test access port architecture for electronic circuits including embedded core having built-in test access port	714/724
46	US 6336182 B1	☒	System and method for utilizing a conditional split for aligning internal operation (IOPs) for dispatch	712/204
47	US 6333916 B1	☒	Wireless communication system, apparatus, and method to communicate using a plurality of communication slots in time division multiple access technique	370/225
48	US 6314330 B1	☒	Single-chip audio system power reduction circuitry and methods	700/94
49	US 6301366 B1	☒	Single-chip audio system mixing circuitry and methods	381/119
50	US 6289418 B1	☒	Address pipelined stack caching method	711/132
51	US 6272624 B1	☒	Method and apparatus for predicting multiple conditional branches	712/239
52	US 6241645 B1	☒	Method and apparatus for changing operating modules	483/1
53	US 6236643 B1	☒	Multiport data switch having variable maximum packet length	370/254
54	US 6201492 B1	☒	Techniques for converting a plurality of channels continuously in an A/D converter	341/155
55	US 6167488 A	☒	Stack caching circuit with overflow/underflow unit	711/132
56	US 6138240 A	☒	Secure general purpose input/output pins for protecting computer system resources	713/202
57	US 6131144 A	☒	Stack caching method with overflow/underflow control using pointers	711/132
58	US 6092798 A	☒	Ticket issuing apparatus	270/52.12
59	US 6047351 A	☒	Jitter free instruction execution	710/266

tional instruction, it is further determined by software of the interrupt handler whether the condition of the conditional instruction is satisfied. Only when this condition is satisfied, a break-interrupt actually occurs.

Thus, like in the 16th embodiment, when the supplied instruction word is a conditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. Even when the supplied instruction word is an unconditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition is satisfied.

In the example shown in FIG. **57**, processing corresponding to each determination section of the scalar processing shown in FIG. **35** is implemented by the function of software. Processing corresponding to each determination section of the VLIW type processor described in each of the embodiments shown in FIGS. **36** and **37** or processing corresponding to each determination section of the parallel processor described in each of the embodiments shown in FIGS. **38** and **39** can also be implemented by the function of software.

For example, to apply the function to the VLIW type processor, processing in steps S5 to S7 in the flow chart of FIG. **57** is replaced by processing in steps S15 to S17 shown in FIG. **49** or processing in steps S25 to S27 shown in FIG. **50**. To apply the function to the parallel processor, processing in steps S5 to S7 in the flow chart of FIG. **57** is replaced by processing in steps S35 to S37 shown in FIG. **52** or processing in steps S45 to S47 shown in FIG. **53**.

However, when the processing is replaced by processing in steps S25 to S27 shown in FIG. **50** or in steps S45 to S47 in FIG. **53**, a breakpoint table shown in FIG. **51** need be used.

21st Embodiment

The 21st embodiment of the present invention will be described next with reference to drawings.

FIG. **58** is a flow chart showing the processing procedure of an interrupt handler according to the 21st embodiment. In FIG. **58**, the same step numbers as in FIG. **54** denote the same processing contents as in FIG. **54**. The breakpoint table has the same construction as in FIG. **55**.

In the 21st embodiment, the 19th embodiment and 20th embodiment are combined. More specifically, in this embodiment, a function of switching between an instruction break mode in which an instruction break occurs when the instruction break generation condition is satisfied and a conditional instruction break mode in which an instruction break occurs when both the instruction break generation condition and the condition of the conditional instruction are satisfied is used. In addition, a function of generating a break-interrupt not only when the condition of a conditional instruction is satisfied but also when the breakpoint target instruction word is an unconditional instruction.

Referring to FIG. **58**, when it is determined in step S3 that an entry corresponding to the break-interrupt generation address is found in the breakpoint table shown in FIG. **55**, the flow advances to step S12. In step S12, it is determined by looking up the breakpoint table shown in FIG. **55** whether the instruction break mode or the conditional instruction break mode is set for the entry.

If the instruction break mode is set, the flow immediately advances to step S9 to execute processing for the instruction break because this mode generates a break-interrupt when the instruction break generation condition is satisfied. If the conditional instruction break mode is set, the flow advances to step S5 because this mode generates a break-interrupt only when th,e condition of the conditional instruction is also satisfied.

It is determined in step S5 whether the breakpoint target instruction word is a conditional instruction, and then it is determined in step S6 whether the instruction word is a conditional instruction. If the breakpoint target instruction word is not a conditional instruction, the flow advances to step S9 to execute processing for the instruction break. If the breakpoint target instruction word is a conditional instruction, the flow advances to step S7 to perform the subsequent part of processing. Processing in the remaining steps is the same as described above in FIG. **54**.

As described above, according to the 21st embodiment, when the conditional instruction break mode is designated, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. Additionally, when the instruction break mode is designated, a break-interrupt can be controlled independently of whether the condition of the conditional instruction is satisfied and in accordance with whether the instruction break generation condition is satisfied.

Furthermore, when the conditional instruction break mode is designated, an d the supplied instruction word is a conditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. Also, even when the supplied instruction word is an unconditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition is satisfied.

In the example shown in FIG. **58**, processing corresponding to each determination section of the scalar processing shown in FIG. **41** is implemented by the function of software. Processing corresponding to each determination section of the VLIW type processor described in each of the embodiments shown in FIGS. **42** and **43** or processing corresponding to each determination section of the parallel processor described in each of the embodiments shown in FIGS. **44** and **45** can also be implemented by the function of software.

For example, to apply the function to the VLIW type processor, processing in steps S5 to S7 in the flow chart of FIG. **58** is replaced by processing in steps S15 to S17 shown in FIG. **49** or processing in steps S25 to S27 shown in FIG. **50**. To apply the function to the parallel processor, processing in steps S5 to S7 in the flow chart of FIG. **58** is replaced by processing in steps S35 to S37 shown in FIG. **52** or processing in steps S45 to S47 shown in FIG. **53**.

However, when the processing is replaced by processing in steps S25 to S27 shown in FIG. **50** or in steps S45 to S47 in FIG. **53**, a breakpoint table shown in FIG. **56** must be used.

22nd Embodiment

The 22nd embodiment of the present invention will be described next with reference to drawings.

In the 18th to 21st embodiments, an application to a so-called instruction breakpoint function has been described, in which the address of an instruction requesting an interrupt is set in a register, and a break-interrupt is generated when the instruction break address set, in this register matches the address of the actually executed instruction.

	Document ID	U	Title	Current OR
60	US 6042478 A	☒	Hand held video game	463/44
61	US 6009474 A	☒	Method and apparatus for re-assigning network addresses to network servers by re-configuring a client host connected thereto	709/245
62	US 5963142 A	☒	Security control for personal computer	340/5.74
63	US 5946262 A	☒	RAM having multiple ports sharing common memory locations	365/230.05
64	US 5928339 A	☒	DMA-transferring stream data apparatus between a memory and ports where a command list includes size and start address of data stored in the memory	710/26
65	US 5923897 A	☒	System for adapter with status and command registers to provide status information to operating system and processor operative to write eject command to command register	710/5
66	US 5901316 A	☒	Float register spill cache method, system, and computer program product	717/158
67	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
68	US 5857087 A	☒	Method of handshaking in a data communications bus	710/305
69	US 5854841 A	☒	Communication system	713/152
70	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24
71	US 5850542 A	☒	Microprocessor instruction hedge-fetching in a multiprediction branch environment	712/235
72	US 5838944 A	☒	System for storing processor register data after a mispredicted branch	712/218
73	US RE35934 E	☒	Semiconductor memory device synchronous with external clock signal for outputting data bits through a small number of data lines	365/189.05
74	US 5826054 A	☒	Compressed Instruction format for use in a VLIW processor	712/213
75	US 5815696 A	☒	Pipeline processor including interrupt control system for accurately perform interrupt processing even applied to VLIW and delay branch instruction in delay slot	712/233
76	US 5809294 A	☒	Parallel processing unit which processes branch instructions without decreased performance when a branch is taken	712/233
77	US 5809258 A	☒	Bus with high gross data transfer rate	710/107
78	US 5802392 A	☒	System for transferring 32-bit double word IDE data sequentially without an intervening instruction by automatically incrementing I/O port address and translating incremented address	710/4
79	US 5787302 A	☒	Software for producing instructions in a compressed format for a VLIW processor	712/24
80	US 5768627 A	☒	External parallel-port device using a timer to measure and adjust data transfer rate	710/60
81	US 5717881 A	☒	Data processing system for processing one and two parcel instructions	712/205

In the 22nd embodiment to be described below, an application to a so-called software breakpoint function will be described, in which an instruction designated at an arbitrary position in a processor is replaced with a breakpoint instruction for an interrupt, and a break-interrupt is generated when the replaced breakpoint instruction is executed during sequential execution of the program.

In the 22nd embodiment, the overall construction of a data processing system (processor) for implementing the software break scheme is the same as that shown in FIG. **4**. Referring to FIG. **4**, when a breakpoint instruction is supplied in executing an instruction supplied from an instruction fetch section **20** by an instruction execution section **30**, the instruction execution section **30** notifies an interrupt control section **40** of the software break-interrupt using an interrupt notification signal **82**.

When receiving the interrupt notification signal **82** from the instruction execution section **30**, the interrupt control section **40** reads out an instruction address **73** at the time of interrupt from a program counter **21** of the instruction fetch section **20** and writes the instruction address **73** in a return address register **52** of a register section **50**. A start address **66** of the interrupt handler for determining the condition of a conditional instruction is supplied to the instruction fetch section **20** and set in the program counter **21**. The interrupt control section **40** also writes the processor state before the interrupt in a previous state register **53** and writes, in a present state register **54**, the processor state that has transited in accordance with the interrupt. Thus, the processor transits from the user state to the supervisor state.

The processor that has transited to the supervisor state executes processing of the interrupt handler sequentially from the start address **66** of the interrupt handler, which is set in the program counter **21**, in accordance with the flow chart shown in FIG. **47**, **49**, **50**, **52**, **53**, **54**, **57**, or **58**.

In these flow charts, in step S4, not error processing for an invalid instruction break but error processing for an invalid software break is performed. In step S9, not processing for an instruction break but processing for a software break is performed. In step S11, not return processing from an instruction break-interrupt but return processing from a software break-interrupt is performed.

As a breakpoint table used in these processing operations, one of the breakpoint tables shown in FIGS. **5**, **59**, **60**, and **61** is used in accordance with the presence/absence of displacement information (DISP) from the head portion;of a long instruction word of a VLIW type processor or a variable-length instruction word of a parallel processor, or the switching function (MODE) between the instruction break mode and the conditional instruction break mode.

In the 22nd embodiment as well, the same effects as described in the 18th to 22nd embodiments can be obtained.

The interrupt control apparatus of each of the above-described embodiments is constructed by a CPU, a MPU, a RAM, or a ROM of a computer and can be implemented by running a program stored in the RAM or ROM. Hence, the apparatus can be implemented by recording a program for causing the computer to execute the above functions on a recording medium such as a CD-ROM and causing the computer to load the program. As a recording medium for recording the program, not a CD-ROM but a floppy disk, a hard disk, a magnetic tape, an optical magnetic disk, or a nonvolatile memory card may be used.

The functions of the above-described embodiments need not always be implemented by causing the computer to execute the supplied program. Such a program is incorporated in the embodiments of the present invention even when the functions of the above-described embodiments are implemented by the program cooperating with the OS (Operating System) or another application software program running on the computer, or the functions of the above-described embodiments are implemented by entirely or partially executing processing of the supplied program by the function expansion board or function expansion unit of the computer.

The above-described embodiments are merely detailed examples for practice of the present invention and do not allow limited interpretation of the technical scope thereof. That is, the present invention may be embodied in various forms without departing from the spirit and scope thereof.

What is claimed is:

1. An interrupt control apparatus having a function of normal interrupt and a function of break-interrupt, said apparatus comprising:

a first information holding section for holding, at the time of a normal interrupt, operation information of a processor before said normal interrupt;

a second information holding section for holding, at the time of a break-interrupt, operation information of said processor before said break-interrupt;

a return operation specifying section for specifying whether a return operation from a normal interrupt state or a return operation from a break-interrupt state is to be performed in returning from an interrupt operation; and

an interrupt return section for re-setting operation information held in said first information holding section or operation information held in said second information holding section in accordance with operation contents specified by said return operation specifying section, and thereby returning the state of said processor from an interrupt operation state to a state before the interrupt.

2. An apparatus according to claim **1**, wherein said second information holding section holds an instruction address before the break-interrupt, to which said processor is to return from the break-interrupt operation state.

3. An apparatus according to claim **2**, wherein said second information holding section further holds the processor state before the break-interrupt.

4. An apparatus according to claim **2**, wherein said second information holding section further holds a factor of the break-interrupt.

5. An apparatus according to claim **2**, wherein said second information holding section further holds the processor state before the break-interrupt and a factor of the break-interrupt.

6. An apparatus according to claim **1**, wherein said return operation specifying section specifies the return operation using flag information which is set at the time of interrupt and shows whether the interrupt is the break-interrupt or not.

7. An apparatus according to claim **1**, wherein said return operation specifying section specifies the return operation in accordance with a value described in an operand of an interrupt return instruction.

8. An apparatus according to claim **1**, wherein said return operation specifying section specifies the return operation in accordance with contents of an instruction field of an interrupt return instruction.

9. An interrupt control method comprising the steps of:

when a normal interrupt occurs, holding operation information of a processor before said normal interrupt in a first information holding section;

	Document ID	U	Title	Current OR
82	US 5706490 A	☒	Method of processing conditional branch instructions in scalar/vector processor	712/234
83	US 5694564 A	☒	Data processing system a method for performing register renaming having back-up capability	712/216
84	US 5689484 A	☒	Auto-changer and method with an optical scanner which distinguishes title information from other information	369/30.3
85	US 5678016 A	☒	Processor and method for managing execution of an instruction which determine subsequent to dispatch if an instruction is subject to serialization	712/216
86	US 5640194 A	☒	Method of multiplexed data reading and visual search suitable for video-on-demand system	725/92
87	US 5627982 A	☒	Apparatus for simultaneously scheduling instructions from plural instruction stream into plural instruction executions units	712/206
88	US 5615331 A	☒	System and method for debugging a computing system	714/9
89	US 5613135 A	☒	Portable computer having dedicated register group and peripheral controller bus between system bus and peripheral controller	710/62
90	US 5563496 A	☒	Battery monitoring and charging control unit	320/128
91	US 5546593 A	☒	Multistream instruction processor able to reduce interlocks by having a wait state for an instruction stream	712/228
92	US 5544311 A	☒	On-chip debug port	714/40
93	US 5539873 A	☒	Picture storage apparatus and graphic engine apparatus	345/502
94	US 5509130 A	☒	Method and apparatus for grouping multiple instructions, issuing grouped instructions simultaneously, and executing grouped instructions in a pipelined processor	712/215
95	US 5491825 A	☒	Microprocessor having a functionally multiplexed input and output terminal	710/305
96	US 5491804 A	☒	Method and apparatus for automatic initialization of pluggable option cards	710/7
97	US 5471593 A	☒	Computer processor with an efficient means of executing many instructions simultaneously	712/235
98	US 5450547 A	☒	Bus interface using pending channel information stored in single circular queue for controlling channels of data transfer within multiple FIFO devices	713/600
99	US 5450409 A	☒	Multiport-multipoint digital data service	370/470
100	US 5442770 A	☒	Triple port cache memory	711/3
101	US 5430851 A	☒	Apparatus for simultaneously scheduling instruction from plural instruction streams into plural instruction execution units	712/212
102	US 5426606 A	☒	Semiconductor memory device synchronous with external clock signal for outputting data bits through a small number of data lines	365/189.05
103	US 5414712 A	☒	Method for transmitting data using a communication interface box	714/712
104	US 5375225 A	☒	System for emulating I/O device requests through status word locations corresponding to respective device addresses having read/write locations and status information	703/25

when a break-interrupt occurs, holding operation information of said processor before said break-interrupt in a second information holding section different from said first information holding section, and setting a flag for showing whether or not the break-interrupt state is set, to the break-interrupt state; and

in returning said processor from the interrupt state to a state before the interrupt, selecting and restoring one of operation information in said first information holding section and operation information in said second information holding section in accordance with a value of said flag.

10. An interrupt control method comprising the steps of:

when a normal interrupt occurs, holding operation information of a processor before said normal interrupt in a first information holding section;

when a break-interrupt occurs, holding operation information of said processor before said break-interrupt in a second information holding section different from said first information holding section; and

in returning said processor from the interrupt state to a state before the interrupt, selecting and restoring one of operation information in said first information holding section and operation information in said second information holding section in accordance with contents of an interrupt return instruction.

11. An interrupt control method for an interrupt control apparatus having a function of normal interrupt and a function of break-interrupt, said method comprising the steps of:

when a break-interrupt occurs, holding at least an instruction address before said break-interrupt, to which a processor is to return from a break-interrupt state, and setting a flag for representing whether or not said break-interrupt state is set, to said break-interrupt state; and

in returning said processor from said break-interrupt state to a state before said break-interrupt, canceling said flag for representing said break-interrupt state, and restoring said instruction address which has been held.

12. A method according to claim **11**, further comprising the steps of:

when said break-interrupt occurs, holding not only said instruction address but also the processor state before said break-interrupt, and in returning said processor from said break-interrupt state to said state before said break-interrupt, restoring said processor state, which has been held.

13. A method according to claim **11**, further comprising the step of:

when said break-interrupt occurs, holding not only said instruction address but also a factor of said break-interrupt.

* * * * *

	Document ID	U	Title	Current OR
105	US 5363485 A	☒	Bus interface having single and multiple channel FIFO devices using pending channel information stored in a circular queue for transfer of information therein	710/113
106	US 5359602 A	☒	Multiport-multipoint digital data service	370/401
107	US 5335326 A	☒	Multichannel FIFO device channel sequencer	710/306
108	US 5303230 A	☒	Fault tolerant communication control processor	370/458
109	US 5276682 A	☒	Medium access technique for LAN systems	370/443
110	US 5199485 A	☒	Air conditioner for motor vehicle having right, left and center temperature controlled vents	165/203
111	US 5177739 A	☒	Multiport - multipoint digital data service	370/449
112	US 5177679 A	☒	Picoprocessor	712/36
113	US 5126944 A	☒	Data processing apparatus for producing in sequence pulses having variable width at output ports	701/103
114	US 5113093 A	☒	Semiconductor integrated circuit with multiple operation	326/16
115	US 5090552 A	☒	Three dimensional sorting apparatus	198/370.04
116	US 5088053 A	☒	Memory controller as for a video signal processor	345/535
117	US 5038320 A	☒	Computer system with automatic initialization of pluggable option cards	710/10
118	US 4958273 A	☒	Multiprocessor system architecture with high availability	712/29
119	US 4888741 A	☒	Memory with cache register interface structure	365/230.05
120	US 4875054 A	☒	Clean air hood for fluid jet printing	347/21
121	US 4868783 A	☒	Dynamic port reconfiguration	710/104
122	US 4827397 A	☒	Microcomputer-based spark ignition gas burner control system	700/81
123	US 4819693 A	☒	Fast operating bistable valve	137/625.4
124	US 4809261 A	☒	Space and time switch for 22 PCM highways	370/374
125	US 4769839 A	☒	Method and device for the transfer of data in a data loop	370/458
126	US 4760553 A	☒	Terminal system configuration tracing method and apparatus	714/45
127	US 4674083 A	☒	Time division multiplexed switching structure for PBX	370/363

rnp

Printed by HPS Server
for
Walk-Up_Printing

Printer: cpk2_2c21_gblrptr
Date: 02/19/04
Time: 13:35:36

Document Listing

Document	Selected Pages	Page Range	Copies
US006032247	22	1 - 22	1
US006381190	9	1 - 9	1
Total (2)	31	-	-

	Document ID	U	Title	Current OR
128	US 4563736 A	☒	Memory architecture for facilitating optimum replaceable unit (ORU) detection and diagnosis	714/42
129	US 4399836 A	☒	Self-contained closed-loop electrically operated valve	137/487.5
130	US 4385206 A	☒	Programmable port sense and control signal preprocessor for a central office switching system	709/244
131	US 4381543 A	☒	Controller port switch arrangement for sharing stored data among different systems	710/316
132	US 4343218 A	☒	Electronic musical instrument	84/625
133	US 4307461 A	☒	Call processor for a satellite communications controller	379/269
134	US 4211895 A	☒	Electronic telephone system with time division multiplexed signalling	370/384
135	US 4205203 A	☒	Methods and apparatus for digitally signaling sounds and tones in a PCM multiplex system	370/525
136	US 4200224 A	☒	Method and system for isolating faults in a microprocessor and a machine controlled by the microprocessor	714/31
137	US 4156932 A	☒	Programmable communications controller	710/3
138	US 4150649 A	☒	Load responsive EGR valve	123/568.29
139	US 4108119 A	☒	Bottom cycle manifold for four-stroke internal combustion engines	123/315
140	US 4103326 A	☒	Time-slicing method and apparatus for disk drive	718/107
141	US 4091455 A	☒	Input/output maintenance access apparatus	714/25
142	US 4090239 A	☒	Interval timer for use in an input/output system	713/502
143	US 4084234 A	☒	Cache write capacity	711/118
144	US 4084232 A	☒	Power confidence system	714/22
145	US 4061880 A	☒	Time-multiplex programmable switching apparatus	370/382
146	US 4050432 A	☒	Fuel injection pump and governor and timing control system therefor	123/502
147	US 4017839 A	☒	Input/output multiplexer security system	710/51
148	US 4006466 A	☒	Programmable interface apparatus and method	710/48
149	US 4000487 A	☒	Steering code generating apparatus for use in an input/output processing system	710/40
150	US 3885103 A	☒	Automatic branch exchange using time division switching	370/378

HPS Trailer Page
for

Walk-Up_Printing

UserID: rnp

Printer: cpk2_2c21_gblrptr

Summary

Document	Pages	Printed	Missed	Copies
US006032247	22	22	0	1
US006381190	9	9	0	1
Total (2)	31	31	0	-

	Document ID	U	Title	Current OR
151	US 3833888 A	☒	GENERAL PURPOSE DIGITAL PROCESSOR FOR TERMINAL DEVICES	710/1
152	US 3815104 A	☒	INFORMATION PROCESSING SYSTEM	710/45
153	US 3629846 A	☒	TIME-VERSUS-LOCATION PATHFINDER FOR A TIME DIVISION SWITCH	711/167
154	US 3596256 A	☐	TRANSACTION COMPUTER SYSTEM HAVING MULTIPLE ACCESS STATIONS	710/45

HPS Trailer Page
for

Walk-Up_Printing

UserID: h

Printer: cpk2_2c21_gblrptr

Summary

Document	Pages	Printed	Missed	Copies
US004312034	68	68	0	1
US004788655	18	18	0	1
US005630157	114	114	0	1
Total (3)	200	200	0	-

	L #	Hits	Search Text	DBs
1	L1	7711	(operation instruction command) near30 ((issu$3 dispatch$3 schedul$3 register) near30 (port slot))	USPAT; US-PGPUB
2	L2	368	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 1	USPAT; US-PGPUB
3	L3	608	(long compound) adj2 instruction and 1	USPAT; US-PGPUB
4	L4	154275	(issu$3 dispatch$3 schedul$3 (port slot)).ab,ti.	USPAT; US-PGPUB
5	L8	969	(operation instruction command) near30 ((issu$3 dispatch$3 schedul$3 register) near30 (port slot))	EPO; JPO; DERWENT; IBM_TDB
6	L11	19	(long compound) adj2 instruction and 8	EPO; JPO; DERWENT; IBM_TDB
7	L9	29	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 8	EPO; JPO; DERWENT; IBM_TDB
8	L5	199	3 and 4	USPAT; US-PGPUB
9	L7	214	2 not 6	USPAT; US-PGPUB
10	L6	154	2 and 4	USPAT; US-PGPUB

h

Printed by HPS Server
for

Walk-Up_Printing

Printer: cpk2_2c21_gblrptr
Date: 02/19/04
Time: 11:16:52

Document Listing

Document	Selected Pages	Page Range	Copies
US004312034	68	1 - 68	1
US004788655	18	1 - 18	1
US005630157	114	1 - 114	1
Total (3)	200	-	-

	L #	Hits	Search Text	DBs
1	L9	106	(operation instruction command) near10 (issue adj1 slot)	USPAT; US-PGPUB

setup of integrated POS registers (112 to 115);

control of data buffer registers by signals DIR, LLDEN, and LHDEN;

generation of read or write input-output commands with signals LIORD and LIOWR;

decoding a read only memory ROM field by generating signal LCSROM, this field being programmable in terms of position and size in the ROM memory space of the BIOS extension defined by POS register (112);

decoding an input-output (IO) field by the LCSIO signal, this field being programmable in terms of position and size in the input-output (IO) space defined in the POS registers (112, 114, 115);

management of READY channel signal (CDCHRDY) to obtain a synchronous cycle at 300 ns or an asynchronous cycle higher than 300 ns with the local input (RDY);

generation of CDSFDBK signal

generation of CDDS16 signal for a 16 bit field;

local inputs of signals MEM816 and IO816, defining the width of the memory or input-output data bus.

For modes 0 and 2, this circuit also allows selection of POS registers (110, 117) by sending signal (LCSPOS) and generating card enable signal (LCDEN). For mode 0, this circuit implements:

the function of management of a DMA channel in the single or burst mode, authorizing a channel by defining the burst size and the arbitration level programmable in register (113);

generation of the DACK signal after participation in an arbitration phase following a DRQ request.

In modes 1 and 2, this circuit implements the memory command generation functions by furnishing signals (LMEMRD) and (LMEMWR) and the random access memory RAM field decode function by furnishing signal (LCSRAM), this field being programmable in terms of position and size in the system memory space by the POS registers (112, 113, 114) in mode 2 and the POS register (113) in mode 1.

This circuit also furnishes the multiplex function of one local interrupt line out of four interrupt levels defined by POS register (102) in mode 2 and by POS register (113) in mode 1. Also, in mode 2, this circuit furnishes the signal generation function (CHCK) and management of bits 6 and 7 of the POS register (115) with local inputs (ICHCK) and (STAT).

In addition, in modes 1 and 3, this circuit furnishes the selection function of the external identification registers by sending signals (OEID0) and (OEID1).

Finally, in mode 3, this circuit adds to the functions of modes 1 and 3 and in common to the four modes, a function of management of two DMA channels in the single mode with internal arbitration and programmable arbitration level in POS register (113) and a signal (DACK0) or (DACK1) generation function after an MCA arbitration phase following a request (DRQ0) or (DRQ1), the fairness mode always being active.

It will be noted that this circuit allows control of generation of the identifiers necessary for operation with an MCA bus by decoding the POS (110) and POS (111) addresses and commanding access to the registers in modes 1, 2, and 3. In the hardwired identifier mode for a card containing no read only memory of the PROM type, the circuit generates signals (OEID0) or (OEID1) with (LA0 at 0 or 1) in order to control the external buffer generating the coupler card identifier.

In mode 0, in a nonhardwired fashion, in the case where the coupler card has a read only memory which furnishes this identifier locally after a period. In this case, the circuit generates an identifier request directed to the local interface and opens the coupler data buffer sending signal (CDPOS) with (LAO-2 at 0 or 1).

Thus, depending on the operating modes selected, this circuit can be used in various applications shown in FIGS. 6 to 9.

FIG. 11 is the schematic diagram of an intelligent interface disk controller card. This card is connected to a bus (1) of the MCA (Micro Channel Architecture) type via a bus arbitration integrated circuit (102) which is the interface between bus (1) and the circuits on the card. For further details on this integrated circuit (102), see the patent application filed by Bull S.A. entitled "MCA Bus Arbitrator Integrated Circuit and Applications of Such a Circuit."

This circuit (102) manages, by means of signal (CSROM) sent over line (623), the accesses to a memory (3) of the EPROM type which contains the BIOS (Basic Input/Output System) part of the disk interface. A coupler circuit (4) physically provides the input-output interface between the central processing unit and a microprocessor circuit (6) as well as the interface between a CACHE memory (5) composed of dynamic random access memories and a disk controller circuit (9). Coupler circuit (4) is connected to disk controller circuit (9) of the type marketed by NEC under number 7282, by two separate data buses, and one bus MD (0:7) (450) connecting coupler circuit (4) of CACHE memory (5), and the other bus CD (0:7) (492) connecting the disk controller both to microprocessor (6) and to coupler circuit (4). Coupler circuit (4) receives the data from MCA bus (1) via bus (41) that interfaces with this MCA bus (1) via buffer registers (11) commanded by signals DIR, LDEN, and HDEN from arbitration circuit (102).

A bus (34) also allows EPROM memory (3) containing the BIOS interface program to be connected with bus (41). A bus (241) connects this bus (41) to interface coupler (2) for the MCA bus. Coupler circuit (4) also receives, via a bus (412), addresses LA (0:15) coming through buffer circuits (612) from address bus A (0:23) (121) which connects MCA bus (1) to interface coupler circuit (102). These buffer circuits (612) are commanded by signal (ADL) from arbitration circuit (102). Signals (MA, OE24, SBHE) also pass through this bus (121). A control bus (120) allows signals (S0, S1, M/IO, DL, CMD) from MCA bus (1) to be received.

Coupler interface circuit (2) transmits via bus (213), interfaced by a buffer register (13), and receives signals ARB (0:3) via bus (210). A control bus (21) receives and transmits signals (TC, BURST, PREEMPT, ARB/-GNT). Interface circuit (2) sends, via control bus (24), signals (CSPOS, CSIO; IORD, IOWR). Circuit (2) receives or transmits via control lines (42), signals (DRQ, DACK). Coupler circuit (4) receives via line (694) the index signal coming from the disk and, via lines (640), signals (IOSTB, MRQ, R/W, REFRQ) coming from microprocessor circuit (6). Microprocessor circuit (6) also receives signals (ITGA) and (ATN) coming from coupler circuit (4) via lines (460). This microprocessor (6) is connected by an address bus CA (0:19) (674), with both coupler circuit (4) and static memory (7), and finally with EPROM memory (8) which contains the card operating program. Coupler circuit (*) sends control signals (WE, RAS0, RAS1, CAS0, CAS1) via link (451).

Disk controller (9) receives from disk connector (90), via lines (95), signals (CMDC, ATN, SCT, DSEL, INDEX, XACK, STSD, DRDY). Likewise, this circuit (9) receives via line (94), the read data signal (RNRZ); via line (93), the

	Document ID	U	Title	Current OR
1	US 20040030839 A1	☐	Cache memory operation	711/137
2	US 20030191928 A1	☒	Multi-way select instructions using accumulated condition codes	712/236
3	US 20030191789 A1	☒	Method and apparatus for implementing single/dual packed multi-way addition instructions having accumulation options	708/670
4	US 20030188143 A1	☒	2N- way MAX/MIN instructions using N-stage 2- way MAX/MIN blocks	712/236
5	US 20030188142 A1	☒	N-wide add-compare-select instruction	712/236
6	US 20030188134 A1	☒	Combined addition/subtraction instruction with a flexible and dynamic source selection mechanism	712/221
7	US 20030154358 A1	☒	Apparatus and method for dispatching very long instruction word having variable length	712/24
8	US 20030149964 A1	☒	Method of executing an interpreter program	717/138
9	US 20030118114 A1	☒	Variable length decoder	375/240.25
10	US 20030105944 A1	☒	Method and apparatus to quiesce a portion of a simultaneous multithreaded central processing unit	712/220
11	US 20030074654 A1	☒	Automatic instruction set architecture generation	717/161
12	US 20030023959 A1	☒	General and efficient method for transforming predicated execution to static speculation	717/151
13	US 20030005261 A1	☒	Method and apparatus for attaching accelerator hardware containing internal state to a processing core	712/35
14	US 20020156999 A1	☒	Mixed-mode hardware multithreading	712/228
15	US 20020144092 A1	☒	Handling of loops in processors	712/217
16	US 20020144078 A1	☒	Address translation	711/203
17	US 20020133784 A1	☒	Automatic design of VLIW processors	716/1

-continued

Pin Number	Type	Symbol	Description
			indicates selection of the connector for access to the POS registers connected to the connector.
10	Input	RDY	Local bus "ready" signal for asynchronous extension of the MCA cycle. The unready status is the low status. This signal must be generated such that the MCA bus signal CDHRDY does not exceed 3 μs.
11–12		NC	Not connected.
13–14	Input	S0 S1	MCA bus status bit. These lines indicate the start of a cycle and its type (read with S1 at low level and S0 at high level, write with S0 at low level and S0 at high level or reserved with S0 and S1 at the same levels).
15	Input	MIO	Memory/input-output cycle. This signal makes a distinction between a memory cycle and an input-output (IO) cycle. The type of cycle, write or read, is defined as a function of the values of S0 and S1. Thus, if M/IO and S1 are at the high level and S0 at the low level, a write memory cycle is run. High level: memory cycle Low level: IO cycle
16	Input	LADL	Address decode latch (or "lock"). This signal allows an address or status decode to be locked to its rising front to ensure proper running of the cycle.
17	Input	IO816	8 or 16 bit input-output (IO). This input determines the given bus width of the input-outputs (IO) which allow or disallow generation of signals CDDS16 and LHDEN for the input-output cycles. This line is hardwired and not a dynamic input. High level: 8 bits Low level: 16 bits
18	Input	MEM816	8 or 16 bit memory. This input determines the memory data bus width allowing or disallowing generation of signals CDDS16 and LHDEN for the memory cycles. This line is hardwired and is not a dynamic input. High level: 8 bits Low level: 16 bits
19	Output	LHDEN	High byte data enable. This low level active output must be connected to the input of an external enable circuit (245). This line is active with input-output (IO) access, memory [access], or in the DMA mode on 16 bits corresponding to the address fields programmed in the POS registers (112 to 115).
20	Output	LIOWR	IO write strobe. This signal, active in the low status, indicates an IO write. Decoding of the S0 S1 status and of MIO is authorized by command signal LCMD.
21	Output	LIORD	IO read strobe. This signal, active at low status, indicates an IO read. Decoding of the S0 S1 status and of MIO is authorized by command signal LCMD.
22	Output	LMWBSTDKL	Mode0: BURST. This high active signal connected to the MCA bus through an open collector inverter gate (F38) indicates transfer of a data block. This line is active after a won arbitration phase,

	Document ID	U	Title	Current OR
18	US 20020120914 A1	☒	Automatic design of VLIW processors	716/17
19	US 20020120831 A1	☒	Stall control	712/219
20	US 20020116598 A1	☒	Computer instruction with instruction fetch control bits	712/207
21	US 20020116567 A1	☒	Efficient I-cache structure to support instructions crossing line boundaries	711/3
22	US 20020091996 A1	☒	Predicated execution of instructions in processors	717/124
23	US 20020087830 A1	☒	Circuit and method for instruction compression and dispersal in wide-issue processors	712/204
24	US 20020083293 A1	☒	Register file circuitry	711/203
25	US 20020056036 A1	☒	Instruction sets for processors	712/209
26	US 20020042909 A1	☒	Retargetable compiling system and method	717/149
27	US 20020035705 A1	☒	Information recording method, information recording device, and information storage medium	714/7
28	US 20010047466 A1	☒	Processors having compressed instructions and methods of compressing instructions for processors	712/226
29	US 20010038422 A1	☒	Device interconnect system using analog line	348/478
30	US 20010034855 A1	☒	Information recording method, information recording device, and information storage medium	714/7
31	US 20010021972 A1	☒	Mapping circuitry and method	712/217
32	US 20010020265 A1	☒	Data processor with multi-command instruction words	712/24
33	US 20010020262 A1	☒	Information recording method, information recording device, and information storage medium	711/4
34	US 20010020261 A1	☒	Information recording method, information recording device, and information storage medium	711/4

-continued

Pin Number	Type	Symbol	Description
			following a DRQ request if the location corresponding to bits 6 and 7 of register POS (113) is not at zero. Mode1 Mode2: LMEMWR, memory write strobe. This signal, active at the low status, indicates memory write. Decoding of statuses S0, status S1, and of MIO authorized by command signal LCMD. Mode3: LDACK1, acknowledge channel 1. This signal, active at low status, indicates an input-output (IO) cycle in DMA mode after an arbitration phase won after a DRQ1 request.
24	Output	LMRDOPREMPT	Mode0 Mode3: OPREMPT, request to MCA. This high active signal connected to the MCA bus through an open collector inverter gate (F38) indicates request of the MCA bus following a request DRQ. This line becomes inactive after a won arbitration phase. Mode1 Mode2: LMEMRD, memory read strobe. This signal, active at low status, indicates a memory read. Decoding of statuses S0, S1, and MIO is authorized by command signal LCMD.
25	Input	CHCKDRQ0	Mode0 Mode3: DRQ0, DMA request in mode0. This input, active at high status, indicates a DMA channel request and triggers an MCA bus preempt.
	Output	CHCKDRQ0	Mode2: CHCK, channel check. This output, active at high status, connected to the MCA bus through an open collector inverter, indicates appearance of a serious error that could disrupt system operation. This line becomes inactive after writing to 1 of bit 7 [of] POS 115. Mode1: not used.
26	Input	014MHZ	Clock at 14.318 MHz. This clock is used for internal arbitration of requests DRQ0 and DRQ1 in modes.
27	Input	ARBSTAT	Mode0 Mode3: Arbitration, ARB/GNT. This high level input indicates an arbitration phase during which the competing priority levels are presented on the MCA bus (ARB0–3). At the time of the descending front of this signal, the MCA bus is allocated to the highest priority that maintains its ARB level, as long as this signal is at the low level; the other competitors have withdrawn their levels. Mode2: STAT, status. This local input reports the presence of an error status. This signal, active in the high status, activates bit 6 of POS 115 if an error is reported by the (CHCK) signal. Mode1: not used.
28	Input	ARB0INTO	Mode0 Mode3: ARB0, arbitration number. This signal from the MCA bus indicates the priority number and allows determination during the arbitration phase (with ARB1–3) of the winner. The ARB0 signal is the least significant bit. Mode1 Mode2: INTO interrupt. This local signal active at the high status indicates appearance of an interrupt.

	Document ID	U	Title	Current OR
35	US 20010 01872 7 A1	☒	Information recording method, information recording device, and information storage medium	711/112
36	US 20010 01690 1 A1	☒	Communicating instruction results in processors and compiling methods for processors	712/217
37	US 66751 92 B2	☒	Temporary halting of thread execution until monitoring of armed events to memory location identified in working registers	718/107
38	US 66586 55 B1	☒	Method of executing an interpreter program	717/139
39	US 66512 22 B2	☒	Automatic design of VLIW processors	716/1
40	US 66474 68 B1	☒	Method and system for optimizing translation buffer recovery after a miss operation within a multi-processor environment	711/147
41	US 66293 12 B1	☒	Programmatic synthesis of a machine description for retargeting a compiler	717/136
42	US 66292 33 B1	☒	Secondary reorder buffer microprocessor	712/218
43	US 66292 31 B1	☒	System and method for efficient register file conversion of denormal numbers between scalar and SIMD formats	712/1
44	US 65811 87 B2	☒	Automatic design of VLIW processors	716/1
45	US 65499 30 B1	☒	Method for scheduling threads in a multithreaded processor	718/104
46	US 65264 21 B1	☒	Method of scheduling garbage collection	707/206
47	US 64937 41 B1	☒	Method and apparatus to quiesce a portion of a simultaneous multithreaded central processing unit	718/107
48	US 64907 16 B1	☒	Automated design of processor instruction units	716/18
49	US 64809 38 B2	☒	Efficient I-cache structure to support instructions crossing line boundaries	711/125
50	US 64635 24 B1	☒	Superscalar processor and method for incrementally issuing store instructions	712/221
51	US 64571 73 B1	☒	Automatic design of VLIW instruction formats	717/149
52	US 64272 35 B1	☒	Method and apparatus for performing prefetching at the critical section level	717/148
53	US 64218 26 B1	☒	Method and apparatus for performing prefetching at the function level	717/161
54	US 64084 28 B1	☒	Automated design of processor systems using feedback from internal measurements of candidate systems	716/17
55	US 63857 57 B1	☒	Auto design of VLIW processors	716/1
56	US 62894 37 B1	☒	Data processing system and method for implementing an efficient out-of-order issue mechanism	712/217

-continued

Pin Number	Type	Symbol	Description
29	Input	LPREMPT	Mode0 Mode3: PREMPT, MCA bus request. This MCA bus signal active at the low status indicates an MCA bus request. Mode1 Mode2: not used.
30	Input	TCCHCKDQ1	Mode0: TC (Terminal Count). This MCA bus signal, active at the low status, indicates the end of a DMA mode exchange. This input is used to interrupt the BURST signal. Mode1: not used. Mode2: ICHCK, error check. This local signal, active at low status, indicates appearance of a serious error. This input is used to generate the CHCK signal as well as bit 7 of POS 105. Mode3: DRQ1, DNA request on channel 1. This local input, active at high status, indicates request for a DMA channel and triggers a preempt of the MCA bus.
31	Input	CHRESET	Channel reset. This MCA bus signal, active at high status, is used when powering on to set up the circuit.
32	Input	LCMD	Command. This signal from the MCA bus is active at low status. This input indicates that the data are valid on the bus. Its rising front indicates end of cycle. This signal is used to authorize input-output (IO) and memory commands as well as data buffer commands.
33–34	Input	MODE0 MODE1	These local signals indicate the operating mode of the circuit. Mode0 Mode1 0 0 Mode0 1 0 Mode1 0 1 Mode2 1 1 Mode3
35	Input	TESTIN	Circuit test input signal. This input must be at the low level in operation.
36	Output	TESTOUT	Circuit test output signal.
37–40	Output	OARB0IT0 OARB1IT1 OARB2IT2 OARB3IT3	Mode0 Mode3: OARB (0–3), arbitration number. Bus arbitration priority level. These 4 bits are connected to MCA bus ARB through open inverter collectors (F38) and are active at the time of an arbitration phase following an MCA bus request. These signals are maintained when an arbitration is won and disappear if it is lost. Mode1 Mode2: IT (0–3) interrupt number. Interrupt lines. These 4 outputs are connected to MCA bus IRQ through open inverter collectors (F38). The INT input is transmitted to multiplexer (103) which sends it to one of the 4 IT outputs (0–3) according to the configuration of bits 6 and 7 (NIT) of POS register (112) in Mode2. These lines are active at the high level.
41	Output	LCSRAMDK0	Mode0 Mode3: LDACK0, acknowledge channel 0. This local output, active at low status, indicates an input-output (IO) cycle in the DMA mode after an arbitration phase won following a DRQ0 request in Mode3. This local output, active at low status, indicates arbitration won following a DRQ0 request in Mode1.

	Document ID	U	Title	Current OR
57	US 6237073 B1	☒	Method for providing virtual memory to physical memory page mapping in a computer operating system that randomly samples state information	711/202
58	US 6195748 B1	☒	Apparatus for sampling instruction execution information in a processor pipeline	712/227
59	US 6189088 B1	☒	Forwarding stored dara fetched for out-of-order load/read operation to over-taken operation read-accessing same memory location	712/216
60	US 6175814 B1	☒	Apparatus for determining the instantaneous average number of instructions processed	702/182
61	US 6163840 A	☒	Method and apparatus for sampling multiple potentially concurrent instructions in a processor pipeline	712/227
62	US 6148396 A	☒	Apparatus for sampling path history in a processor pipeline	712/227
63	US 6141675 A	☒	Method and apparatus for custom operations	708/706
64	US 6131152 A	☒	Planar cache layout and instruction stream therefor	712/24
65	US 6122722 A	☒	VLIW processor with less instruction issue slots than functional units	712/24
66	US 6119075 A	☒	Method for estimating statistics of properties of interactions processed by a processor pipeline	702/186
67	US 6092180 A	☒	Method for measuring latencies by randomly selected sampling of the instructions while the instruction are executed	712/200
68	US 6076154 A	☒	VLIW processor has different functional units operating on commands of different widths	712/24
69	US 6070009 A	☒	Method for estimating execution rates of program execution paths	717/130
70	US 6055619 A	☒	Circuits, system, and methods for processing multiple data streams	712/36
71	US 6044451 A	☒	VLIW processor with write control unit for allowing less write buses than functional units	712/24
72	US 6009483 A	☒	System for dynamically setting and modifying internal functions externally of a data processing apparatus by storing and restoring a state in progress of internal functions being executed	710/36
73	US 6002880 A	☒	VLIW processor with less instruction issue slots than functional units	712/24
74	US 6000044 A	☒	Apparatus for randomly sampling instructions in a processor pipeline	714/47
75	US 5974537 A	☒	Guard bits in a VLIW instruction control routing of operations to functional units allowing two issue slots to specify the same functional unit	712/215
76	US 5966530 A	☒	Structure and method for instruction boundary machine state restoration	712/244
77	US 5964867 A	☒	Method for inserting memory prefetch operations based on measured latencies in a program optimizer	712/219
78	US 5963744 A	☒	Method and apparatus for custom operations of a processor	712/9

-continued

Pin Number	Type	Symbol	Description
			This line becomes inactive at the next arbitration. Mode1 Mode2: LCSRAM, chip select RAM. This local output active at low status indicates selection of the RAM field programmed in bits 2 of register POS 112 and 0 of register 114 for modes 1 and 2 and (on the one hand) register 113 in mode 2 or (on the other hand) bits 3 to 7 of register 113 in mode 1. Addresses and valid statuses are locked by LCMD.
44	Output	LCSIO	Chip select Input-Outputs. This local output, active at low status, indicates selection of the input-output (IO) field programmed in the POS registers, on the one hand whose size in modes 0, 1, and 3 is indicated by bits 1, 2 (IOA) of register 112 and whose position is indicated by register 114 and bits 0–4 of register 115, and on the other hand is validated (enabled) in mode 2 by bit 1 of register 112, and whose position is indicated by bits 0–4 of register 115 and 2–7 of register 114.
45	Output	LCSROM	Chip select ROM. This local output, active at low status, indicates selection of the ROM field programmed in fields (SEG ROM) of bits (3–5) of register 112 and bits 6–7 (EROM) of register 112 in modes 0, 1, and 3 or of bit 1 (VROM) of register 114 in mode 2. Valid addresses and status reads are locked by LCMD.
46	Output	LCSPOEID0	Mode0 Mode2: LCSPOS - POS register selection. This local output, active at low status, indicates selection of POS registers 110 to 117. The LSETUP signal is locked by LADL. Mode1 Mode3: OEID0, identifier output enable (output enable ID0). This local output active at low status indicates reading of register POS 110.
47	Output	LCDENOEID1	Mode0 Mode2: LCDEN card enable. This local output active at low status indicates that the card is active (bit 0 POS 112). Mode1 Mode3: OEID1, identifier ID1 output enable (output enable ID1). This local output active at low status indicates reading of register POS 111.
48	Input	BHE	System byte high enable. This signal from the MCA bus is used to validate the high byte when accessing a 16-bit system.
49–60	Input	A0–A11	MCA bus addresses.
68–79		A12–A23	These bits, of which A0 is the least significant bit and A23 the most significant, are used to decode the memory and input-output (IO) resources.
61	Input	WS1ARB3	Mode0 Mode3: ARB3, arbitration level. This MCA bus signal indicates the priority level and enables the winner to be determined during the arbitration phase (with ARB0–ARB2). ARB3 is the most significant bit.
	Output	WS1ARB3	Mode2: WS1, size of memory window. This local output indicates with

	Document ID	U	Title	Current OR
79	US 5931939 A	☒	Read crossbar elimination in a VLIW processor	712/24
80	US 5923872 A	☒	Apparatus for sampling instruction operand or result values in a processor pipeline	712/244
81	US 5923863 A	☒	Software mechanism for accurately handling exceptions generated by instructions scheduled speculatively due to branch elimination	712/216
82	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
83	US 5862399 A	☒	Write control unit	712/24
84	US 5862398 A	☒	Compiler generating swizzled instructions usable in a simplified cache layout	712/24
85	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24
86	US 5838940 A	☒	Method and apparatus for rotating active instructions in a parallel data processor	712/216
87	US 5826054 A	☒	Compressed Instruction format for use in a VLIW processor	712/213
88	US 5809250 A	☒	Method for estimating statistics of properties of instructions processed by a processor pipeline	702/186
89	US 5787302 A	☒	Software for producing instructions in a compressed format for a VLIW processor	712/24
90	US 5751985 A	☒	Processor structure and method for tracking instruction status to maintain precise state	712/218
91	US 5694564 A	☒	Data processing system a method for performing register renaming having back-up capability	712/216
92	US 5673426 A	☒	Processor structure and method for tracking floating-point exceptions	712/244
93	US 5673408 A	☒	Processor structure and method for renamable trap-stack	712/216
94	US 5659721 A	☒	Processor structure and method for checkpointing instructions to maintain precise state	712/228
95	US 5651115 A	☒	Processor structure and method for watchpoint of plural simultaneous unresolved branch evaluation	712/239
96	US 5651124 A	☒	Processor structure and method for aggressively scheduling long latency instructions including load/store instructions while maintaining precise state	712/215
97	US 5649136 A	☒	Processor structure and method for maintaining and restoring precise state at any instruction boundary	712/244
98	US 5644742 A	☒	Processor structure and method for a time-out checkpoint	712/244
99	US 5634023 A	☒	Software mechanism for accurately handling exceptions generated by speculatively scheduled instructions	712/244
100	US 5627981 A	☒	Software mechanism for accurately handling exceptions generated by instructions scheduled speculatively due to branch elimination	712/235
101	US 5598546 A	☒	Dual-architecture super-scalar pipeline	712/209

-continued

Pin Number	Type	Symbol	Description
			bits WS0 and M116 the size of the memory used by the card. This size is defined in mode 2 by bits 6, 7 (WSIZE) of register 113 and bit 0 (M116) of register 114. M116 = 0: in first megabyte WS0 WS1 Size 1 1 64 kB 0 1 32 kB 1 0 16 kB M116 = 1: over 1 megabyte WS0 WS1 Size 1 1 1 MB 0 1 512 MB 1 0 256 MB
62	Input	WS0ARB2	Mode0 Mode3: ARB2, arbitration level. This signal from the MCA bus indicates the priority level and enables the winner to be determined during the arbitration phase (with ARB0-1-3).
	Output	WS0ARB2	Mode2: WS0, memory window size. This local output indicates, with bits WS1 and M116, the size of the memory used by the card. See table WS1 ARB3.
63	Input	ARB1M116	Mode0 Mode1 Mode3: ARB1, arbitration level. This signal from the MCA bus indicates the priority level and allows the winner to be determined during the arbitration phase (with ARB0-2-3).
	Output	ARB1M116	Mode2: M116. This local output indicates, as seen above, the position of the memory used in the memory space of the system. See TABLE WS1 ARB3.
65	Output	CDSFDBK	Card selected feedback. This output active at high status connected to the MCA bus through an inverter (F04) indicates selection of a memory or input-output (IO) zone used by the card.
66	Output	CDDS16	Card data size 16. This output active at high status connected to the MCA bus through an inverter (F04) indicates selection of a memory or input-output (IO) zone of 16 bits.
67	Output	CDCHRDY	Channel ready. This output, connected to the MCA bus through an inverter (F04), indicates an unready state at high status. This line can be generated by programming bit 5 of POS register 115 and maintained by the RDY input.
80	Output	LLDEN	Low byte data enable. This local output, active at the low level, indicates low access of a local input-output (IO) or memory resource. This line must be connected to the enable chip of an external buffer (245).
81	Output	DIR	Direction. This local output indicates the direction of data transfers currently under way. A low level indicates a read and a high level, a write. This line must be connected to the direction [sic] of an external buffer (245).
1 43	Input	VDD	5V
2 22 42 64	Input	VSS	Ground

	Document ID	U	Title	Current OR
102	US 5561846 A	☒	Base station and channel monitoring method	370/337
103	US 5542109 A	☒	Address tracking and branch resolution in a processor with multiple execution pipelines and instruction stream discontinuities	712/234
104	US 5349677 A	☒	Apparatus for calculating delay when executing vector tailgating instructions and using delay to facilitate simultaneous reading of operands from and writing of results to same vector register	712/4
105	US 4984156 A	☒	Automatic checkin apparatus	705/5
106	US 3832973 A	☒	APPARATUS FOR THE PRODUCTION OF A MULTI-LAYER SHEET MATERIAL OF MICROPOROUS STRUCTURE	118/50

FIG. 6A shows in greater detail the logic blocks of decode circuit (106) which is composed mainly of a logic block of POS registers 112 to 115, a logic block DPS (1060) for commanding the POS registers, and a logic block (1061) DCS generating selection signals, either from input-outputs or from random access memory RAM or from read only memory ROM as a function of the addresses presented on the address bus.

FIG. 6B is the detailed logic diagram of the POS registers composed of flip-flops whose outputs DPOSXY indicate the number by X, and can be loaded from data bus ID0–ID7 in synchronization with signals LWPOS2, LWPOS3, LWPOS4, and LWPOS5 selecting each of the registers in the write or read modes.

FIG. 6C shows the multiplex logic enabling selection, in the read mode, of any of the four paths constituted by outputs DPOSXY of each of the four registers of which each of the significant bits is sent to a four-input multiplexer and an output ODY, Y being the corresponding number of bits.

FIG. 6D shows the general logic of the POS register selection signals, particularly signals LWPOS2 to LWPOS5 which enable the POS registers and signals SEL 0, SEL 1 which command the multiplexers to be selected or set up. The various signals are generated by decoding addresses IA0 to IA2 of address bus MCA and signals LRSETUP, LRADL, LRESET 1, LCYIOW, LCYIOR, CYIOR and MODE 02.

FIGS. 6E and 6F are the logic diagrams of the circuits that select inputs-outputs by the CSIO signal, random access memory RAM by the CSRAM signal, and read only memory ROM by the LCSROM signal.

These signals are generated by processing the various MIO signals from the MCA bus and MEM816, IO816 from the card and indicating the bus size from these signals, for example by selecting inputs-outputs obtained by comparing addresses A3 to A15 with bits 1 to 7 of the POS4 register and bits 0 to 4 of the POS5 register. Likewise, comparison of addresses A16 to A23 with bits 0 to 2 of the POS3 register forming the word SEGROM in mode 0 or with bits 3 to 5 of the POS2 register in mode 2 and bits 6 and 7 of register POS2 in mode 02 allow generation of the read only memory ROM selection signal by intermediate signal ADROM.

FIG. 6G is the circuit selection logic diagram that selects the RAM memory. In mode 1 this logic makes a comparison between low address bits A14 to A16 furnished by the MCA bus with bits 3 to 5 of the POS3 register constituting the SEGRAM information to form an ADRAM address enable signal. A RAM memory enable signal ENRAM is formed either by combining bits 6 and 7 of the POS3 register in the case of mode 1 operation or by bit 2 of the POS2 register in the case of mode 2 operation of the circuit. These signals produce the ENRAM signal by addition. The address enable signal is also to be produced in mode 2 by comparing address bits A14 to A19 furnished by the address bus with bits 0 to 5 of the POS3 register or by comparing bits A18 to A23 furnished by the address bus with the same bits of the POS3 register, this comparison being made according to the value of the MADE24 signal indicating extension of the addresses to 16 megabits.

FIG. 7A shows the breakdown of command logic circuit 105 into two logic blocks (1050, 1051). The first (1050) furnishes the commands from the various elements that may be associated with this arbitration circuit such as a memory, inputs-outputs, or buffer registers. This circuit 105 also has a logic block (1051) BUF forming the buffer selection signals, which, as will be seen below in the applications, may be associated with this arbitration circuit.

FIG. 7B shows production of the ILHDEN and ILLDEN signals from external signals IO816 and MEM816 as LIOWR, MWRBSIDK1, CS10, CSRAM, CSROM and from a signal indicating operation in mode 1 or 2 (mode 1, 2).

FIG. 7C is the detailed logic diagram of block (1050) which produces outputs MWIRBSTDK1 as a function of the modes used and the BURST OPRMPT and DACK1 signals and the MRDOPRMIT signal, also as a function of modes 1 or 2 or mode 0 and 3. These circuits also produce internal signals from inputs SO to S1 indicating the start of a cycle, with MIO indicating whether the cycle is a memory or an input-output cycle, and LADL which ensures that the cycle is running properly by means of the address decode latch (lock). This information is used to generate internal signals LCYMEM indicating a memory cycle, CYIOR indicating an input-output read cycle, and LCYIOR, LCYIOW indicating an input-output write cycle.

FIG. 8A is the breakdown of an arbitration logic circuit (104) into a logic block (1040) handling interrupts, a logic block (1041) measuring arbitration, and a logic block (1042) generating the BURST signal.

FIG. 8B shows the detailed logic diagram of part of logic block 1041 producing, from signals from arbitration level[s] RARB0 to RARB3 coming from the MCA bus and bits 2 to 5 of the POS2 register in mode 0 or bits 0 to 3 or 4 to 7 in mode 3 depending on the arbitration channel enabled by signal ENARB1 or ENARB0, signals OARB0 to OARB3.

FIG. 8C is the detailed logic schematic of the part of logic block 1041 that allows signals ENARB0 to ENARB1 to be produced as well as other arbitration signals by using mode signals, acknowledge signals LDACK1, and request signals on bus LOPRMPT.

FIG. 8D shows the detailed logic diagram of logic block IT (1040) generating the interrupt levels in the arbitration procedure bearing in mind, on the one hand, arbitration numbers OARB0 to OARB3 furnished by arbitration circuit DMA and the values of bits 0 and 1 of the POS3 register in mode 1 constituted by the NIT information and by bits 6 and 7 of register POS2 in mode 2.

FIG. 8E is the detailed logic diagram of logic block (1042) generating the BURST signal. This signal is produced from bits 4 and 5 of the POS3 register and from signals CDEN, LBCMD, LRPREMPT, LARB, ITCCMCKDD1, LWON1, LCYIOL, IDRQO in order to satisfy the DMA cycle with a burst in mode 0 as shown in FIG. 19.

FIG. 9 represents the detailed logic diagram of mode decode circuit (108) that, from signals RMOD0 and RMOD1 available at input pins 3334, allow these signals to be decoded in the form of four signals MODE 0, MODE 1, MODE 2, MODE 3 and additional signals MODE 12, MODE 02, MODE 03 necessary for exploiting and selecting certain information contained in the POS registers by means of the circuits described above.

FIG. 10 is the detailed logic diagram of the circuit generating the ready signal upon reception of the IRDY signal to form the ICDRDY signal intended for the MCA bus by using signals indicating a memory cycle LADROM, LADRAM or an input-output cycle LADIO, as well as by using signals selecting the random access memory or inputs-outputs, CSRAM and CSIO respectively, and by using bit 5 of the POS5 register constituting the RDY information. This circuit also uses signals RMIO and LWON1.

Thus, the integrated circuit represented, once installed on a card, permits the following functions depending on the information contained in the registers, whatever the operating mode used:

	L #	Hits	Search Text	DBs
1	L1	7711	(operation instruction command) near30 ((issu$3 dispatch$3 schedul$3 register) near30 (port slot))	USPAT; US-PGPUB
2	L2	368	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 1	USPAT; US-PGPUB
3	L3	608	(long compound) adj2 instruction and 1	USPAT; US-PGPUB
4	L4	154275	(issu$3 dispatch$3 schedul$3 (port slot)).ab,ti.	USPAT; US-PGPUB
5	L8	969	(operation instruction command) near30 ((issu$3 dispatch$3 schedul$3 register) near30 (port slot))	EPO; JPO; DERWENT; IBM_TDB
6	L11	19	(long compound) adj2 instruction and 8	EPO; JPO; DERWENT; IBM_TDB
7	L9	29	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 8	EPO; JPO; DERWENT; IBM_TDB
8	L5	199	3 and 4	USPAT; US-PGPUB
9	L7	214	2 not 6	USPAT; US-PGPUB
10	L6	154	2 and 4	USPAT; US-PGPUB
11	L18	1241	((issu$3 dispatch$3 schedul$3) and (port slot)).ab,ti.	USPAT; US-PGPUB
12	L21	4730	(operation instruction command) near10 ((issu$3 dispatch$3 schedul$3 register) near10 (port slot)) and (register and (microprocess$3 process$3 comput$3))	USPAT; US-PGPUB
13	L20	35	(operation instruction command) near10 ((issu$3 dispatch$3 schedul$3) near10 (port slot)) and (register and (comput$3 microprocess$3 process$3))	EPO; JPO; DERWENT; IBM_TDB
14	L23	138	18 and 21	USPAT; US-PGPUB

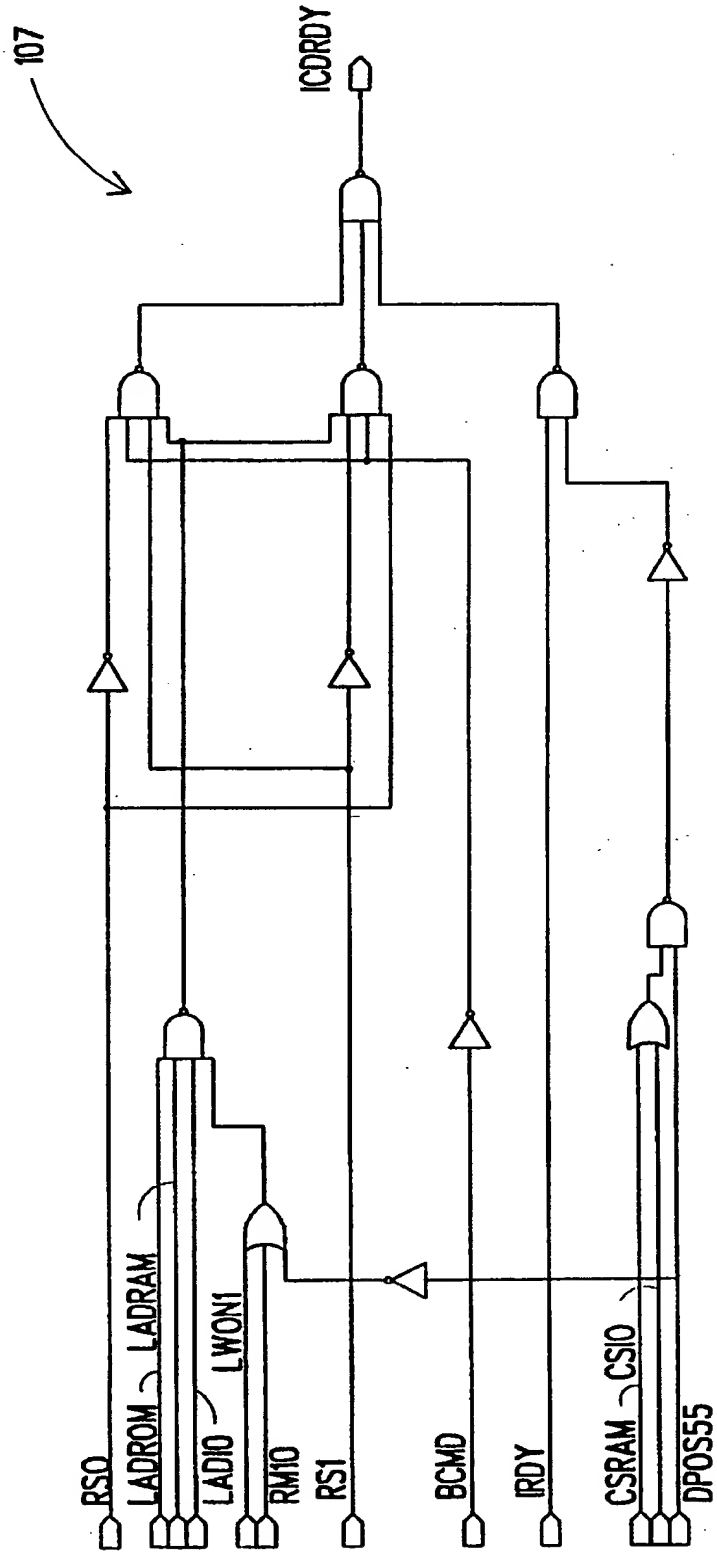
107
ICDRDY
RS0
LADROM
LADRAM
LADIO
LWON1
RM10
RS1
BCMD
IRDY
CSRAM
CS10
DPOS55

FIG. 10

	Document ID	U	Title	Current OR
1	JP 20030 37572 A	☐	SCHEDULING SYSTEM	
2	JP 20012 65593 A	☒	INFORMATION PROCESSOR	
3	JP 09180 032 A	☒	CARD PROCESSOR AND CARD PROCESSING SYSTEM USING THE SAME	
4	JP 08095 956 A	☒	VECTOR PROCESSOR	
5	JP 05158 968 A	☒	INSTRUCTION ISSUE CONTROL DEVICE	
6	JP 02236 649 A	☒	MEMORY CONTROL SYSTEM	
7	JP 62296 243 A	☒	MICROCOMPUTER	
8	WO 98137 55 A2	☒	READ CROSSBAR ELIMINATION IN A VLIW PROCESSOR	
9	EP 72533 4 A1	☒	Executing speculative parallel instruction threads	
10	EP 60592 7 A1	☒	Improved very long instruction word processor architecture.	
11	GB 22666 05 A	☒	Microprocessor having a run/stop pin for accessing an idle mode	
12	NNRD4 20138	☒	Efficient Scheduling and Operand Renaming of Groups of Instructions	
13	NA930 6163	☒	Dealer Instruction Processing Unit Governor Decoders	
14	NN930 5313	☒	Multisequencing a Single Instruction Stream using the Parameter of Occupancy to Moderate Complexity	
15	NN921 2323	☒	Means for Generating a Pass A20 SignalfFrom a Single Register.	
16	WO 20030 88038 A	☒	Multi-issue processor e.g. super scalar processor for real time application, has location in holdable registers in which one set of issue slots is different from location of holdable registers in another set of slots	
17	US 20020 08331 3 A	☒	Data processing apparatus has register file with access ports, functional unit and instruction issue unit	
18	US 20020 01393 7 A	☒	Basic block operations scheduling method for explicit parallel instruction computing architecture, involves scheduling fixed number of ready operations having lowest economy priority in subsequent time slots	
19	US 62860 94 B	☒	Determination of dispatch slot in processing system involves allowing instructions to be directed to specific decode slots and follow dispatch constraints without examination of instruction	
20	US 61483 96 A	☒	State information collection apparatus for computer system, samples selected state information stored in shift register and additional state information about executed instruction, simultaneously during sampling	
21	US 61120 19 A	☒	Distributed instruction queue for superscaler microprocessor, reads all source operands from external storage unit when decoded instruction is issued, but the result value is not forwarded from functional unit	

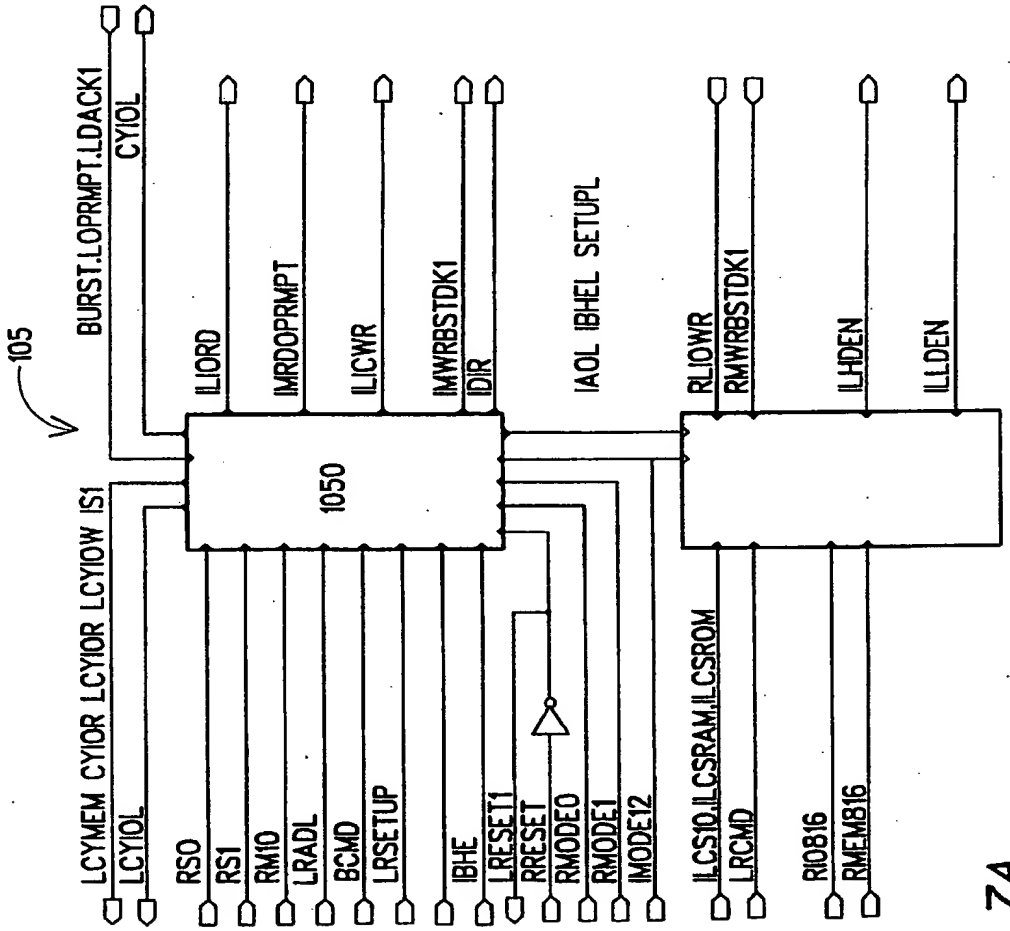
105
BURST.LOPRMPT.LDACK1
CY1OL
ILIORD
IMRDOPRMPT
ILICWR
IMWRBSTDK1
IDIR
IAOL IBHEL SETUPL
RLIOWR
RMWRBSTDK1
ILHDEN
ILLDEN
1050
LCYMEM CYIOR LCYIOR LCYIOW IS1
LCY1OL
RS0
RS1
RM1O
LRADL
BCMD
LRSETUP
IBHE
LRESET1
RRESET
RMODE0
RMODE1
IMODE12
ILCS1O.ILCSRAM.ILCSROM
LRCMD
RIO816
RMEM816

FIG. 7A

	Document ID	U	Title	Current OR
22	US 6035389 A	☒	Apparatus with clock giving pulses and electronic hardware having several rows and one or more ports for scheduling instructions with different latencies for computers and processors has each row recording separate latency vector	
23	US 5983335 A	☒	Multiple out-of-order instructions issuing and execution mechanism in superscalar machine e.g. IBM system	
24	US 5944810 A	☒	Execution results retiring device for superscalar processor	
25	US 5903740 A	☒	Retire instruction apparatus of superscalar microprocessor	
26	RD 420138 A	☒	Greedy instructions scheduling and operand renaming for computer program execution - uses speculative execution and register renaming on single entry multiple blocks in linear time, based on resource availability vectors	
27	US 5870614 A	☒	Read crossbar simplification or elimination for very long instruction word processors - functional units are assigned to particular slots in the instruction issue register so the number of slots is less than number of units, and by associating a read port with each slot the read crossbar either simplified or eliminated	
28	US 5787302 A	☒	VLIW processor using compressed instructions with instruction issue register - byte aligns variable length instructions, branch targets are uncompressed with format bits showing how many issue slots are used in following instruction, NOPS not stored in memory, compresses individual operations based on features	
29	US 5574935 A	☒	Multi-port re-order buffer for superscalar processor - out-of-order dispatch logic dispatches instructions issued to execution units in superscalar execution cluster, re-order buffer stores entries, which includes source data, corresponding to instructions issued	
30	EP 725334 A	☒	CPU in computer for executing parallel threads of instructions - executes sequence of instructions using single instruction control with single program counter and shared set of architecturally visible machine state to determine which threads of instructions can be executed in parallel	
31	US 5481743 A	☒	Minimal instruction set computer system with multiple instruction issuing - has memory with bidirectional data port and instruction buffer for storing multiple instructions to be executed in parallel by multiple processing units	
32	EP 632370 A	☒	Pipelined data processing system - has three stages coupled in series, in which each stage can terminate operation while holding information, except for final stage which is unable to hold information over two time slots	
33	EP 564847 A	☒	Massively parallel processor for image, multi-media and general purpose computing - has processing element unit processor array structure with single and dual processing elements, using finite difference method of solving differential equations	
34	EP 352935 A	☒	Pipelined data processing system - has units interconnected by parameter files providing slots for allocation to instructions in pipeline until successfully completed	
35	DE 3540753 A	☐	Fast dynamic RAM for data processing - entire-data in refresh register is saved in special register with its own column decoder	

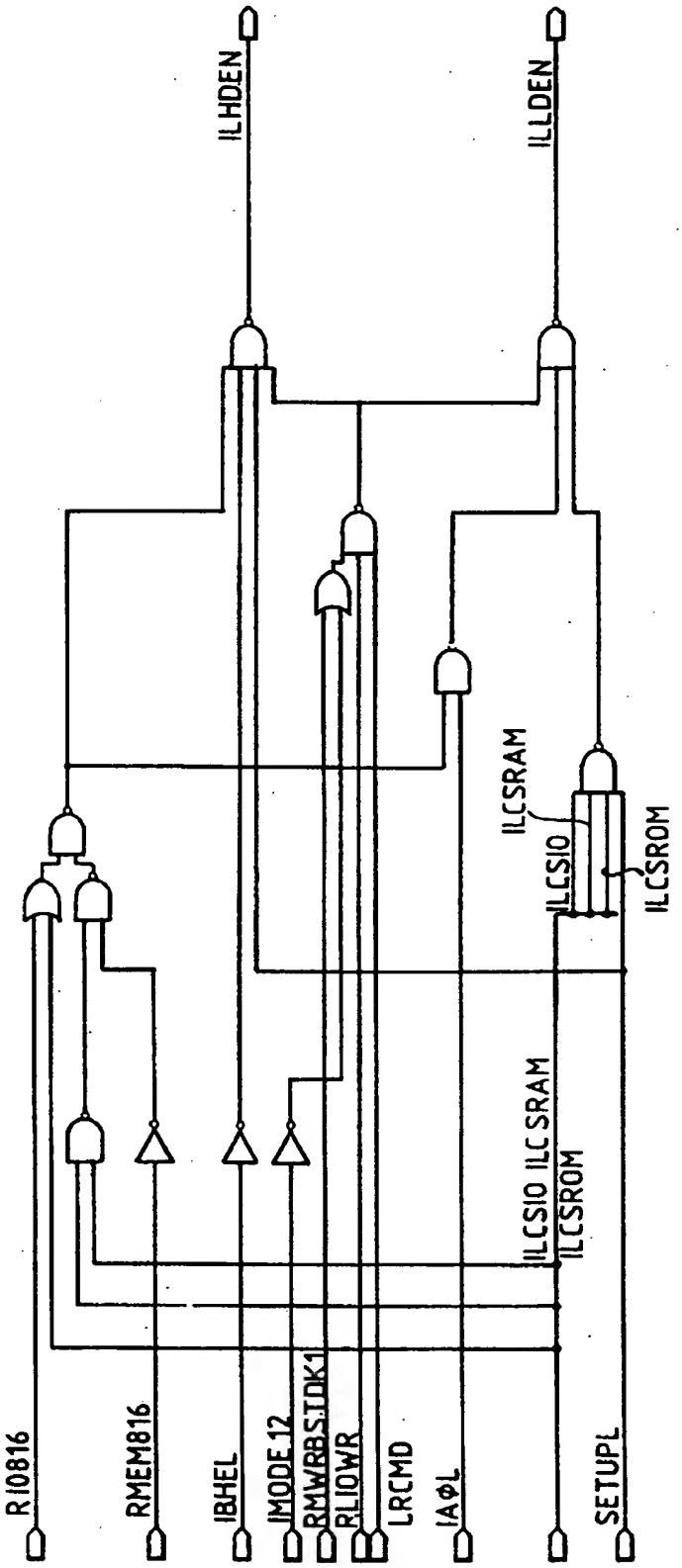
ILHDEN
ILLDEN
ILCSRAM
ILCSIO
ILCSROM
ILCSIO ILC SRAM
ILCSROM
R/IO816
RMEM816
IBHEL
IMODE12
RMWRBSIDK1
RLIOWR
LRCMD
IAΦL
SETUPL

FIG. 7B

	Document ID	U	Title	Current OR
1	US 20040030816 A1	☐	DMA scheduling mechanism	710/52
2	US 20040028071 A1	☒	Apparatus and method for managing variable-sized data slots with timestamp counters within a TDMA frame	370/442
3	US 20040008972 A1	☒	Personal TV receiver (PTR) with program recommendation forwarding function	386/83
4	US 20040008713 A1	☒	System and method for packet transmission from fragmented buffer	370/428
5	US 20040006729 A1	☒	Hierarchical test methodology for multi-core chips	714/733
6	US 20030202517 A1	☒	Apparatus for controlling packet output	370/395.4
7	US 20030200539 A1	☒	Function unit based finite state automata data structure, transitions and methods for making the same	717/161
8	US 20030128712 A1	☒	Packet communication apparatus and controlling method thereof	370/412
9	US 20030120974 A1	☒	Programable multi-port memory bist with compact microcode	714/31
10	US 20030112817 A1	☒	Methods and apparatus for differentiated services over a packet-based network	370/413
11	US 20030088479 A1	☒	Online scheduling system	705/26
12	US 20030046326 A1	☒	Method for creating a schedule, apparatus for creating a schedule, and computer-program for creating a schedule	718/102
13	US 20030043761 A1	☒	Channel structures and protocol for asset tracking satellite communications links	370/319
14	US 20030005260 A1	☒	Superscalar RISC instruction scheduling	712/23
15	US 20020191583 A1	☒	Slot cycle assignment within a communication system	370/345
16	US 20020184330 A1	☒	Shared memory multiprocessor expansion port for multi-node systems	709/213
17	US 20020167955 A1	☒	Packet transfer device and packet transfer method adaptive to a large number of input ports	370/411

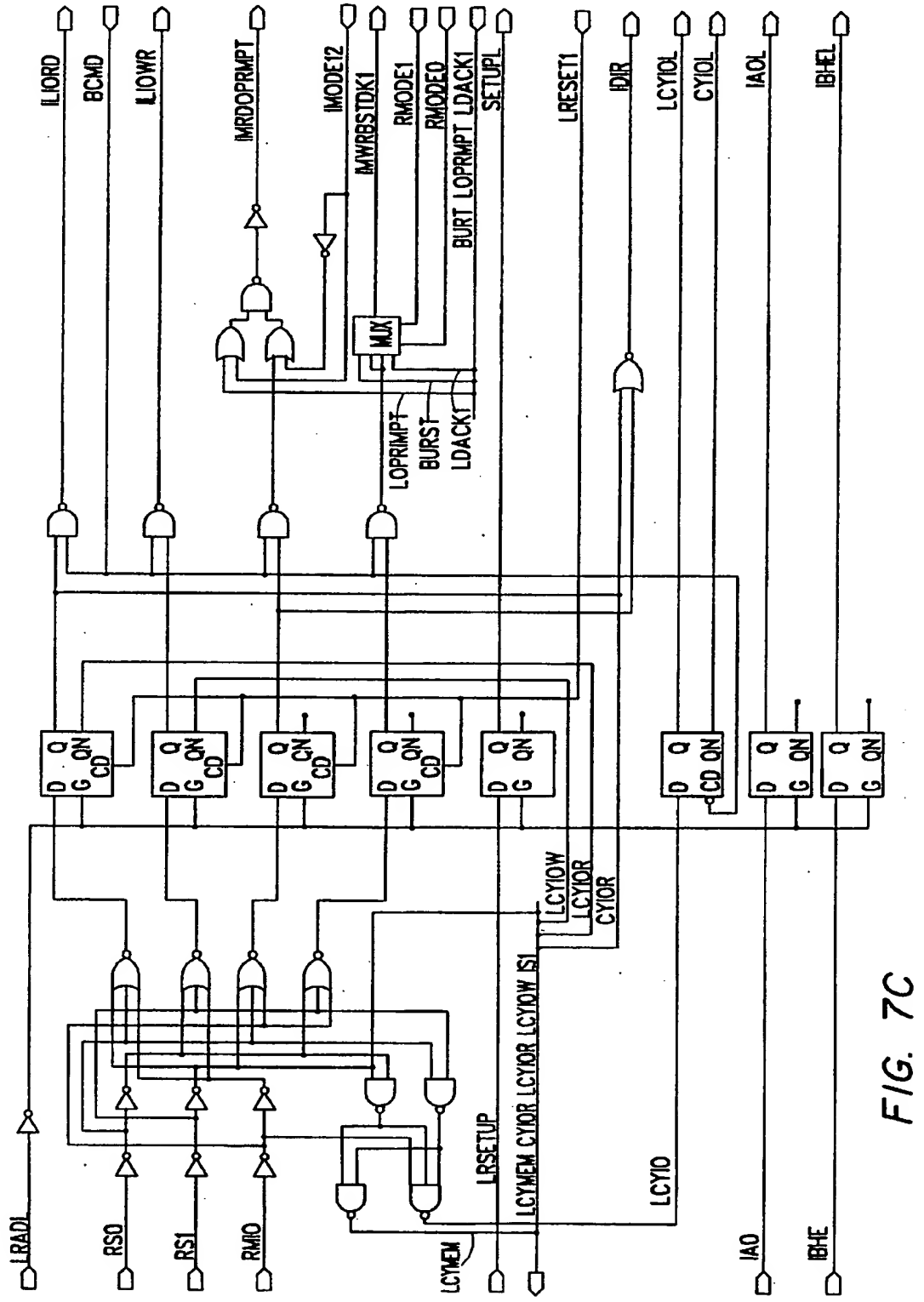
ILIORD
BCMD
LIOWR
IMRDOPRMPT
IMODE12
IMWRBSTDK1
RMODE1
RMODE0
BURT LOPRMPT LDACK1
SETUPL
LRESET1
IDIR
LCYIOL
CYIOL
IAOL
IBHEL
MUX
LOPRIMPT
BURST
LDACK1
LCYIOW
LCYIOR
CYIOR
LRADI
RSO
RS1
RMO
LRSETUP
LCYMEM CYIOR LCYIOR LCYIOW IS1
LCYIO
LCYMEM
IAO
IBHE

FIG. 7C

	Document ID	U	Title	Current OR
18	US 20020136230 A1	☒	Scheduler for a packet routing and switching system	370/416
19	US 20020059426 A1	☒	Technique for assigning schedule resources to multiple ports in correct proportions	709/226
20	US 20020039368 A1	☒	Method and apparatus for preventing undesirable packet download with pending read/write operations in data packet processing	370/412
21	US 20020019927 A1	☒	Apparatus for issuing an instruction to a suitable issue destination	712/214
22	US 20020013937 A1	☒	Register economy heuristic for a cycle driven multiple issue instruction scheduler	717/160
23	US 20010049756 A1	☒	Transport convergence sub-system with shared resources for multiport xDSL system	710/33
24	US 20010023479 A1	☒	Information processing unit, and exception processing method for specific application-purpose operation instruction	712/209
25	US 20010020282 A1	☒	External storage	714/9
26	US 6697073 B1	☒	Image processing system	345/501
27	US 6691221 B2	☒	Loading previously dispatched slots in multiple instruction dispatch buffer before dispatching remaining slots for parallel execution	712/215
28	US 6662294 B1	☒	Converting short branches to predicated instructions	712/226
29	US 6661774 B1	☒	System and method for traffic shaping packet-based signals	370/230.1
30	US 6629233 B1	☒	Secondary reorder buffer microprocessor	712/218
31	US 6611512 B1	☒	Apparatus and method for scheduling correlation operations of a DS-CDMA shared correlator	370/342
32	US 6549538 B1	☒	Computer method and apparatus for managing network ports cluster-wide using a lookaside list	370/395.52
33	US 6546476 B1	☒	Read/write timing for maximum utilization of bi-directional read/write bus	711/167
34	US 6490248 B1	☒	Packet transfer device and packet transfer method adaptive to a large number of input ports	370/229
35	US 6442597 B1	☒	Providing global coherence in SMP systems using response combination block coupled to address switch connecting node controllers to memory	709/214
36	US 6427189 B1	☒	Multiple issue algorithm with over subscription avoidance feature to get high bandwidth through cache pipeline	711/122
37	US 6359891 B1	☒	Asynchronous transfer mode cell processing system with scoreboard scheduling	370/398

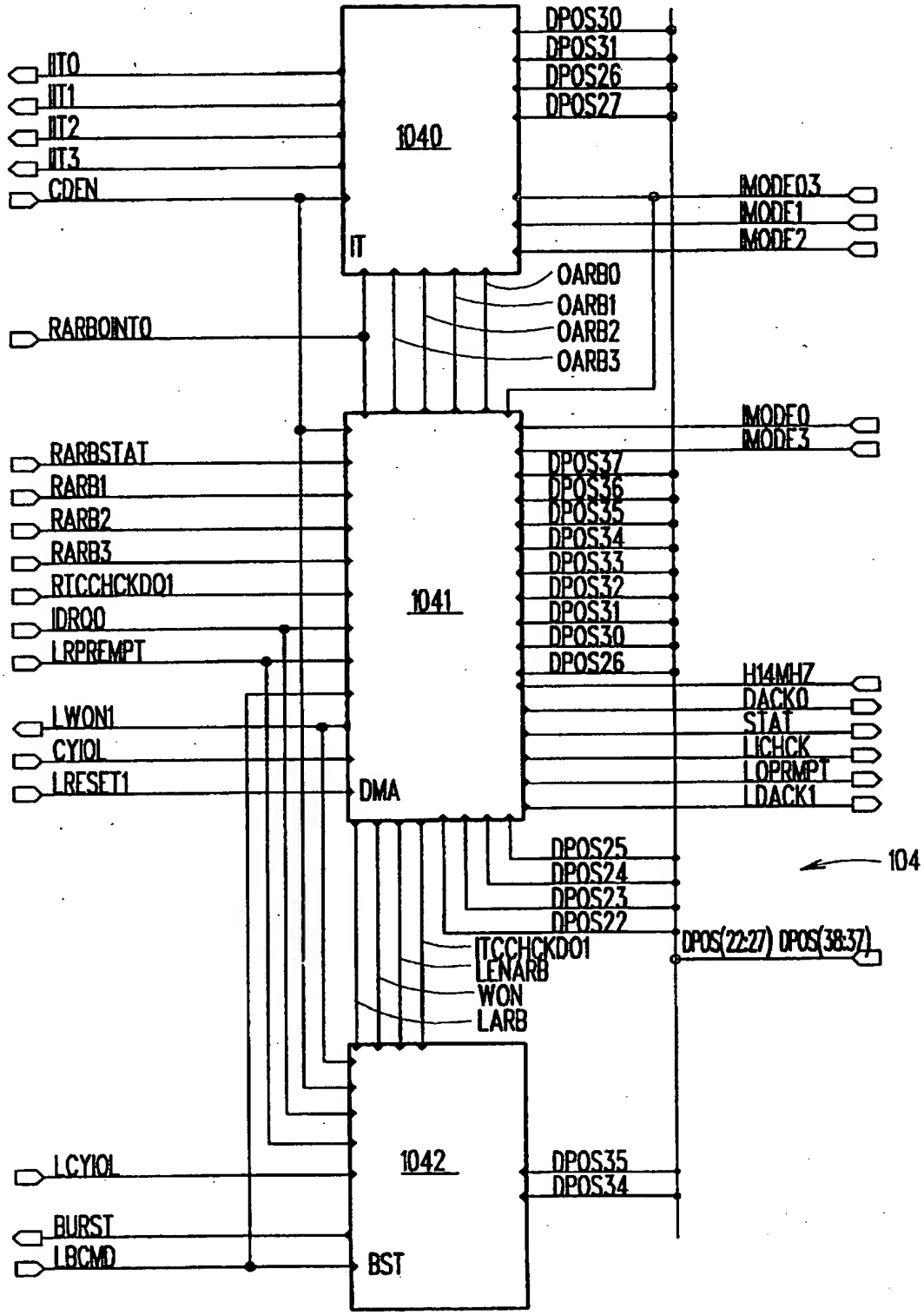
1040
IT
IIT0
IIT1
IIT2
IIT3
CDEN
RARBOINT0
DPOS30
DPOS31
DPOS26
DPOS27
IMODE03
IMODE1
IMODE2
OARB0
OARB1
OARB2
OARB3
1041
DMA
RARBSTAT
RARB1
RARB2
RARB3
RTCCHCKDO1
IDRQ0
LRPREMPT
LWON1
CYI0L
LRESET1
IMODE0
IMODE3
DPOS37
DPOS36
DPOS35
DPOS34
DPOS33
DPOS32
DPOS31
DPOS30
DPOS26
H14MHZ
IDACK0
STAT
LICHCK
LOPRMPT
IDACK1
DPOS25
DPOS24
DPOS23
DPOS22
DPOS(22:27) DPOS(38:37)
104
ITCCHCKDO1
LENARB
WON
LARB
1042
BST
LCYI0L
BURST
LBCMD
DPOS35
DPOS34

FIG. 8A

	Document ID	U	Title	Current OR
38	US 6351802 B1	☒	Method and apparatus for constructing a pre-scheduled instruction cache	712/215
39	US 6336182 B1	☒	System and method for utilizing a conditional split for aligning internal operation (IOPs) for dispatch	712/204
40	US 6336156 B1	☒	Increased speed initialization using dynamic slot allocation	710/45
41	US 6289433 B1	☒	Superscalar RISC instruction scheduling	712/23
42	US 6286094 B1	☒	Method and system for optimizing the fetching of dispatch groups in a superscalar processor	712/213
43	US 6272542 B1	☒	Method and apparatus for managing data pushed asynchronously to a pervasive computing client	709/224
44	US 6263416 B1	☒	Method for reducing number of register file ports in a wide instruction issue processor	712/23
45	US 6192461 B1	☒	Method and apparatus for facilitating multiple storage instruction completions in a superscalar processor during a single clock cycle	712/23
46	US 6189089 B1	☒	Apparatus and method for retiring instructions in excess of the number of accessible write ports	712/218
47	US 6170001 B1	☒	System for transfering format data from format register to memory wherein format data indicating the distribution of single or double precision data type in the register bank	708/495
48	US 6128303 A	☒	Asynchronous transfer mode cell processing system with scoreboard scheduling	370/398
49	US 6122722 A	☒	VLIW processor with less instruction issue slots than functional units	712/24
50	US 6088788 A	☒	Background completion of instruction and associated fetch request in a multithread processor	712/205
51	US 6088774 A	☒	Read/write timing for maximum utilization of bidirectional read/write bus	711/167
52	US 6088772 A	☒	Method and apparatus for improving system performance when reordering commands	711/158
53	US 6076146 A	☒	Cache holding register for delayed update of a cache line into an instruction cache	711/125
54	US 6065110 A	☒	Method and apparatus for loading an instruction buffer of a processor capable of out-of-order instruction issue	712/217
55	US 6052795 A	☒	Recovery method and system for continued I/O processing upon a controller failure	714/3
56	US 6052751 A	☒	Method and apparatus for changing the number of access slots into a memory	710/107
57	US 6044451 A	☒	VLIW processor with write control unit for allowing less write buses than functional units	712/24
58	US 6038396 A	☒	Compiling apparatus and method for a VLIW system computer and a recording medium for storing compile execution programs	717/161
59	US 6035389 A	☒	Scheduling instructions with different latencies	712/216
60	US 6002880 A	☒	VLIW processor with less instruction issue slots than functional units	712/24

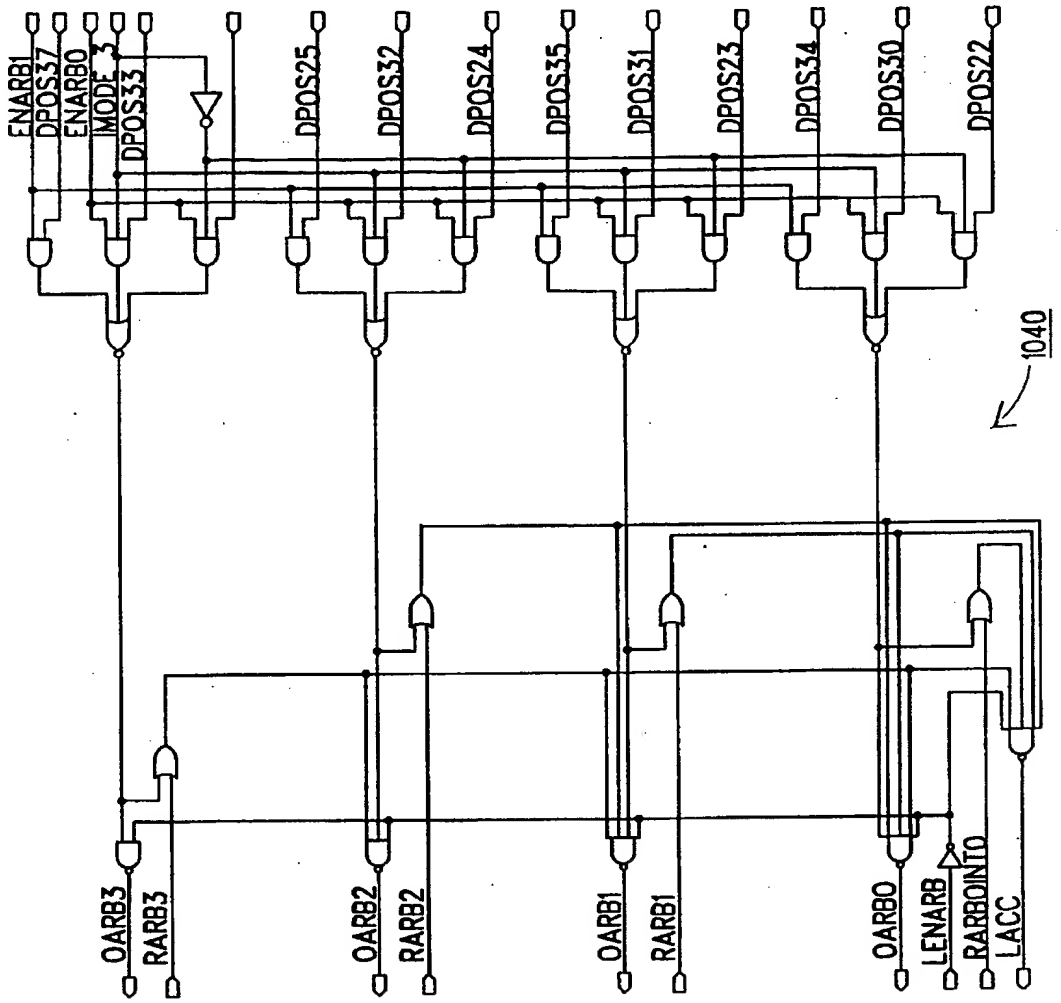
ENARB1
DPOS37
ENARB0
MODE_32
DPOS33
DPOS25
DPOS32
DPOS24
DPOS35
DPOS31
DPOS23
DPOS34
DPOS30
DPOS22
1040
OARB3
RARB3
OARB2
RARB2
OARB1
RARB1
OARB0
LENARB
RARBOINTO
LACC

FIG. 8B

	Document ID	U	Title	Current OR
61	US 5983321 A	☒	Cache holding register for receiving instruction packets and for providing the instruction packets to a predecode unit and instruction cache	711/125
62	US 5982783 A	☒	Switch distribution via an intermediary switching network	370/395.6
63	US 5978902 A	☒	Debug interface including operating system access of a serial/parallel debug port	712/227
64	US 5974537 A	☒	Guard bits in a VLIW instruction control routing of operations to functional units allowing two issue slots to specify the same functional unit	712/215
65	US 5974526 A	☒	Superscalar RISC instruction scheduling	712/23
66	US 5956321 A	☒	Stream scheduling system for real time stream server	370/230
67	US 5903740 A	☒	Apparatus and method for retiring instructions in excess of the number of accessible write ports	712/217
68	US 5887174 A	☒	System, method, and program product for instruction scheduling in the presence of hardware lookahead accomplished by the rescheduling of idle slots	717/161
69	US 5883901 A	☒	Communications system including synchronization information for timing upstream transmission of data and ability to vary slot duration	370/508
70	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
71	US 5870577 A	☒	System and method for dispatching two instructions to the same execution unit in a single cycle	712/215
72	US 5862399 A	☒	Write control unit	712/24
73	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24
74	US 5842036 A	☒	Circuit and method for scheduling instructions by predicting future availability of resources required for execution	712/23
75	US 5826054 A	☒	Compressed Instruction format for use in a VLIW processor	712/213
76	US 5812811 A	☒	Executing speculative parallel instructions threads with forking and inter-thread communication	712/216
77	US 5809550 A	☒	Method and apparatus for pushing a cacheable memory access operation onto a bus controller queue while determining if the cacheable memory access operation hits a cache	711/167
78	US 5809325 A	☒	Circuit and method for scheduling instructions by predicting future availability of resources required for execution	712/32
79	US 5799163 A	☒	Opportunistic operand forwarding to minimize register file read ports	712/205
80	US 5796930 A	☒	System architecture for processing and transporting page-map or bit-map data to a raster print engine	358/1.17
81	US 5790817 A	☒	Configurable digital wireless and wired communications system architecture for implementing baseband functionality	710/311
82	US 5787483 A	☒	High-speed data communications modem	711/158
83	US 5787302 A	☒	Software for producing instructions in a compressed format for a VLIW processor	712/24

H1 4MHZ
IMODE3
IDRO
ITCCHICKDO1
RTCCHICKDO1
LICHCK
LDACK1
LRESET1
IDRO0
H26MHZ
LRESET1
LRPREMPT
IMODE0
DPOS26
RARBSTAT
LACC
LWON1
LRESET1
CYYOL
IMODE0
H14MHZ
ZH28MHZ
DACK0
ENARB0
LDACK1
ENARB1
LACC
MODE03
CDEN
LGAGNE
CDEN
LENARB
LBCMD
CDEN
LOPRMPT
LARB
STAT
WON
LWON1

FIG. 8C

	Document ID	U	Title	Current OR
84	US 5784597 A	☒	Communications network system including acknowledgement indicating successful receipt of request for reserved communication slots and start time for said reserved communication slots	713/401
85	US 5777928 A	☒	Multi-port register	365/189.05
86	US 5761475 A	☒	Computer processor having a register file with reduced read and/or write port bandwidth	712/218
87	US 5758117 A	☒	Method and system for efficiently utilizing rename buffers to reduce dispatch unit stalls in a superscalar processor	712/217
88	US 5748631 A	☒	Asynchronous transfer mode cell processing system with multiple cell source multiplexing	370/398
89	US 5737624 A	☒	Superscalar risc instruction scheduling	712/23
90	US 5721722 A	☒	FA controller and data processing method therefor	700/2
91	US 5710941 A	☒	System for substituting protected mode hard disk driver for real mode driver by trapping test transfers to verify matching geometric translation	710/14
92	US 5694564 A	☒	Data processing system a method for performing register renaming having back-up capability	712/216
93	US 5689674 A	☒	Method and apparatus for binding instructions to dispatch ports of a reservation station	712/217
94	US 5680402 A	☒	Priority broadcast and multi-cast for unbuffered multi-stage networks	370/498
95	US 5673427 A	☒	Packing valid micro operations received from a parallel decoder into adjacent locations of an output queue	712/245
96	US 5630096 A	☒	Controller for a synchronous DRAM that maximizes throughput by allowing memory requests and commands to be issued out of order	711/154
97	US 5619502 A	☒	Static and dynamic scheduling in an asynchronous transfer mode communication network	370/397
98	US 5615331 A	☒	System and method for debugging a computing system	714/9
99	US 5588126 A	☒	Methods and apparatus for fordwarding buffered store data on an out-of-order execution computer system	712/200
100	US 5581709 A	☒	Multiple computer system using I/O port adaptor to selectively route transaction packets to host or shared I/O device	710/38
101	US 5579473 A	☒	Interface controller for frame buffer random access memory devices	345/557
102	US 5574935 A	☒	Superscalar processor with a multi-port reorder buffer	712/23
103	US 5555432 A	☒	Circuit and method for scheduling instructions by predicting future availability of resources required for execution	712/23
104	US 5546593 A	☒	Multistream instruction processor able to reduce interlocks by having a wait state for an instruction stream	712/228
105	US 5532844 A	☒	Image data transferring system and method	358/468

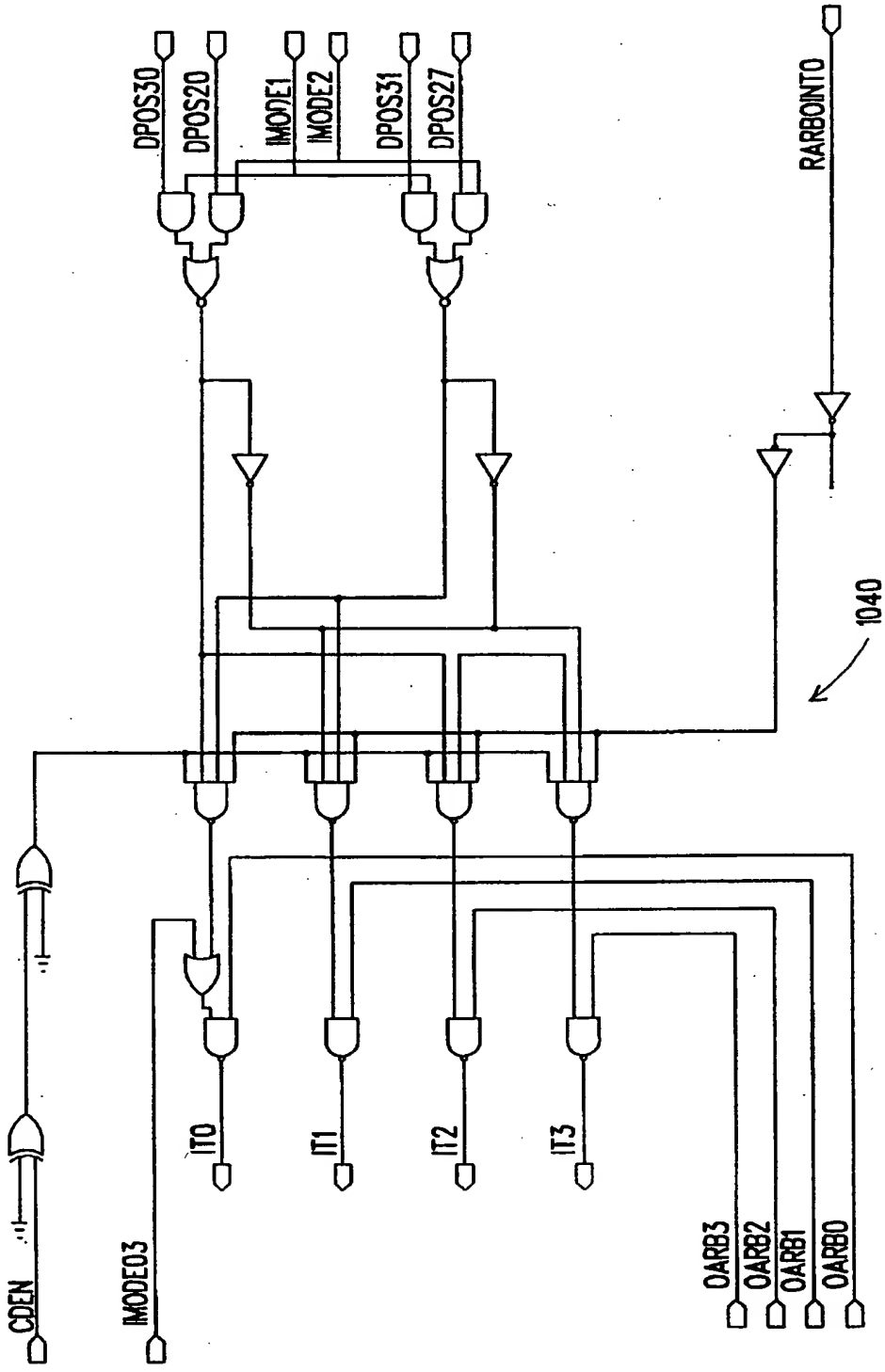
DPOS30
DPOS20
IMODE1
IMODE2
DPOS31
DPOS27
RARBOINT0
1040
IT0
IT1
IT2
IT3
OARB3
OARB2
OARB1
OARB0
CDEN
IMODE03

FIG. 8D

	Document ID	U	Title	Current OR
106	US 5528584 A	☒	High performance path allocation system and method with fairness insurance mechanism for a fiber optic switch	370/254
107	US 5524250 A	☒	Central processing unit for processing a plurality of threads using dedicated general purpose registers and masque register for providing access to the registers	712/228
108	US 5517555 A	☒	Real time information system for cellular telephones	455/408
109	US 5497499 A	☒	Superscalar risc instruction scheduling	712/217
110	US 5488728 A	☒	Microprocessor having a run/stop pin for accessing an idle mode	710/200
111	US 5461407 A	☒	Marking method and apparatus using gas entrained abrasive particles	347/82
112	US 5448702 A	☒	Adapters with descriptor queue management capability	710/100
113	US 5442305 A	☒	Active bus termination device	326/30
114	US 5420987 A	☒	Method and apparatus for configuring a selected adapter unit on a common bus in the presence of other adapter units	710/10
115	US 5386562 A	☒	Circular scheduling method and apparatus for executing computer programs by moving independent instructions out of a loop	717/160
116	US 5357519 A	☒	Diagnostic system	714/25
117	US 5289531 A	☒	Remote scheduling of appointments with interactivety using a caller's unit	379/93.23
118	US H001291 H	☒	Microprocessor in which multiple instructions are executed in one clock cycle by providing separate machine bus access to a register file for different types of instructions	712/23
119	US 5276682 A	☒	Medium access technique for LAN systems	370/443
120	US 5247671 A	☒	Scalable schedules for serial communications controller in data processing systems	718/103
121	US 5237572 A	☒	Active remote module for the attachment of user equipments to a communication processing unit	370/463
122	US 5185737 A	☒	Method and apparatus for cyclic reservation multiple access in a communications system	370/449
123	US 5173898 A	☒	Multiple-access control for a communication system with order pad passing	370/440
124	US 5146589 A	☒	Refresh control for dynamic memory in multiple processor system	714/3
125	US 5117490 A	☒	Pipelined data processor with parameter files for passing parameters between pipeline units	712/218
126	US 5093813 A	☒	Multiple mode electronic scheduler	368/10
127	US 4893234 A	☒	Multi-processor including data flow accelerator module	712/27
128	US 4727475 A	☒	Self-configuring modular computer system with automatic address initialization	710/104

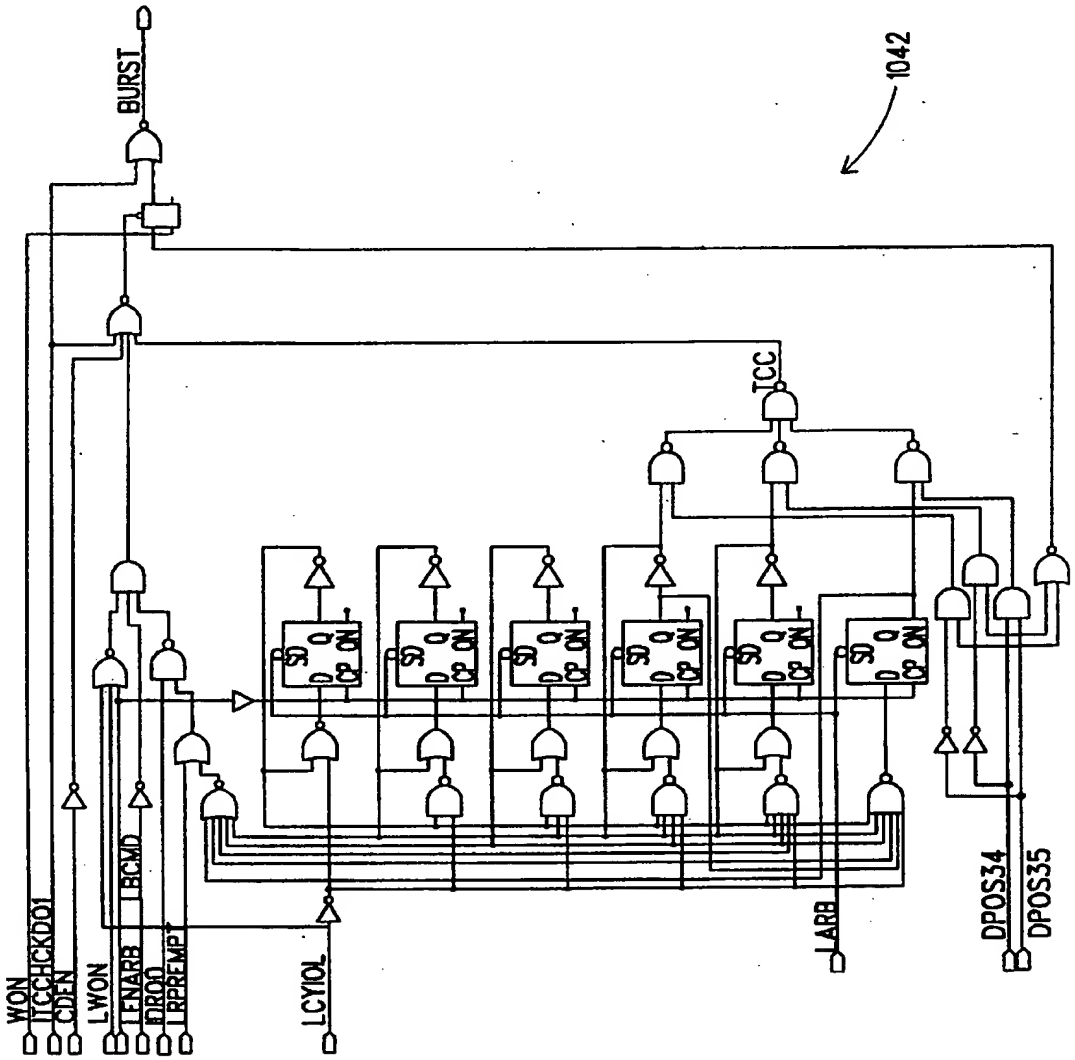
1042
BURST
ICC
SD
CP
QN
WON
ITCCHCKD01
CDEN
LWON
LBCMD
LENARB
DBOD
LRPREMPT
LCYIOL
LARB
DPOS34
DPOS35

FIG. 8E

	Document ID	U	Title	Current OR
129	US 4698567 A	☒	Ribbon deck motor control	318/480
130	US 4688212 A	☒	Centralized image responsive telephone time slot interchange system	370/360
131	US 4651317 A	☒	Time division multiplex data transmission system	370/216
132	US 4597075 A	☒	Modular switching network for telecommunication system	370/366
133	US 4387427 A	☒	Hardware scheduler/dispatcher for data processing system	718/102
134	US 4348743 A	☒	Single chip MOS/LSI microcomputer with binary timer	713/502
135	US 4325120 A	☒	Data processing system	711/202
136	US 4253146 A	☒	Module for coupling computer-processors	709/226
137	US 4205203 A	☒	Methods and apparatus for digitally signaling sounds and tones in a PCM multiplex system	370/525
138	US 4087643 A	☐	Time division multiplexed PABX communication switching system	370/212

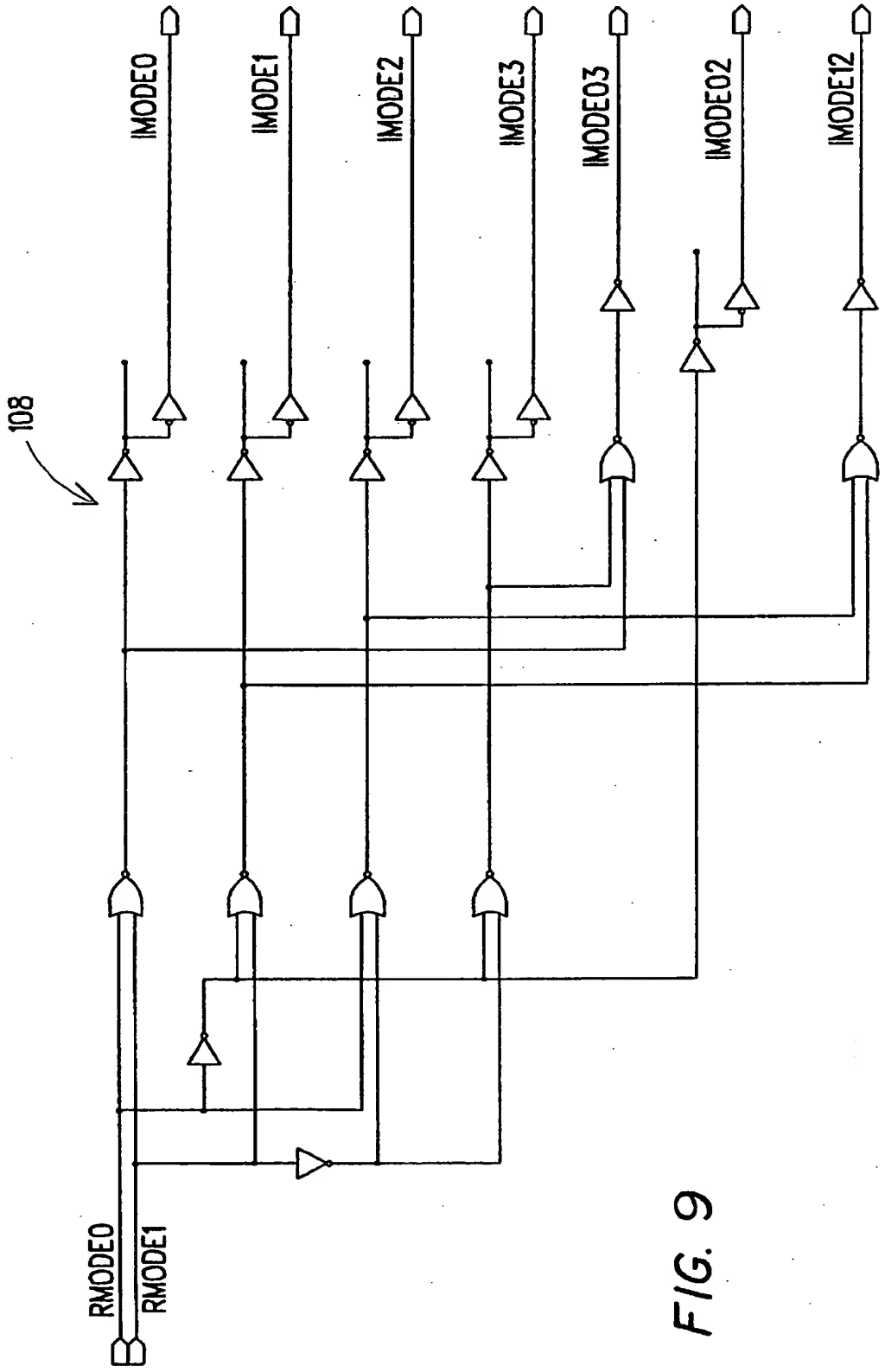
IMODE0
IMODE1
IMODE2
IMODE3
IMODE03
IMODE02
IMODE12
108
RMODE0
RMODE1

FIG. 9

	L #	Hits	Search Text	DBs
1	L1	7711	(operation instruction command) near30 ((issu$3 dispatch$3 schedul$3 register) near30 (port slot))	USPAT; US-PGPUB
2	L2	368	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 1	USPAT; US-PGPUB
3	L3	608	(long compound) adj2 instruction and 1	USPAT; US-PGPUB
4	L4	154275	(issu$3 dispatch$3 schedul$3 (port slot)).ab,ti.	USPAT; US-PGPUB
5	L6	154	2 and 4	USPAT; US-PGPUB
6	L8	969	(operation instruction command) near30 ((issu$3 dispatch$3 schedul$3 register) near30 (port slot))	EPO; JPO; DERWENT; IBM_TDB
7	L11	19	(long compound) adj2 instruction and 8	EPO; JPO; DERWENT; IBM_TDB
8	L9	29	(replac$3 substitut$3 modifi$5 alter$3 chang$3) near20 8	EPO; JPO; DERWENT; IBM_TDB
9	L5	199	3 and 4	USPAT; US-PGPUB
10	L7	214	2 not 6	USPAT; US-PGPUB

interrupt from a program counter **21** of an instruction fetch section **20** and writes the instruction address **73** in a return address register **52** of a register section **50**. A start address **66** of the interrupt handler for determining the condition of the conditional instruction is supplied to the instruction fetch section **20** and set in the program counter **21**. The interrupt control section **40** also writes the processor state before the interrupt in a previous state register **53** and writes, in a present state register **54**, the processor state that has transited in accordance with the interrupt. Thus, the processor transits from the user state to the supervisor state.

The processor that has transited to the supervisor state executes processing of the interrupt handler sequentially from the start address **66** of the interrupt handler, which is set in the program counter **21**.

FIG. **47** is a flow chart showing the processing procedure of the interrupt handler according to the 18th embodiment. In the 18th embodiment, interrupt control based on determination of the condition of a conditional instruction, which is done by the hardware mechanism in the scalar processor of the 10th embodiment, is performed by the function of software shown in the flow chart of FIG. **47**.

Referring to FIG. **47**, the context is saved in step S1. More specifically, the value of the register section **50** used by the interrupt handler is written in the memory **10**. In step S2, an entry corresponding to a break-interrupt generation address is obtained by looking up, e.g., the breakpoint table of an instruction execution section **30** shown in FIG. **3**. More specifically, a breakpoint table as shown in FIG. **48** is set and held in the instruction execution section **30** in advance. Referring to FIG. **48**, a column "VALID" represents whether the breakpoint operation is valid. The value "1" means a breakpoint operation valid state while the value "0" means a breakpoint operation invalid state. A column "ADDRESS" holds the target address of a breakpoint at which execution is to be stopped.

In step S2, an entry where one of instruction addresses stored in the address registers **24***a* of the breakpoint registers 24_{-0} to 24_{-n} shown in FIG. **3** or an instruction address as the factor of the break-interrupt is present is obtained from entries #0 to #n by looking up the column "ADDRESS" of the breakpoint table shown in FIG. **48**.

In step S3, it is determined whether an entry corresponding to the break-interrupt generation address is found in the breakpoint table. If NO in step S3, it is determined that the instruction break is invalid. The flow advances to step S4 to execute error processing for the invalid instruction break. If YES in step S3, the flow advances to step S5.

In step S5, whether the breakpoint target instruction word (instruction word as the factor of the interrupt notification signal **67**) is a conditional instruction. In step S6, it is determined whether the breakpoint target instruction word is a conditional instruction. If the instruction word is not a conditional instruction, the flow jumps to step S10 without performing processing for the instruction break.

If the breakpoint target instruction word is a conditional instruction, the flow advances to step S7. In step S7, it is determined by referring to the condition register **51** whether the condition of the breakpoint target instruction word as a conditional instruction is satisfied. In step S8, it is determined whether the condition of the conditional instruction is satisfied. If the condition is not satisfied, the flow jumps to step S10 without performing processing for the instruction break.

If the condition of the conditional instruction is satisfied, the flow advances to step S9 to execute processing for the instruction break. More specifically, in accordance with the sequentially incremented value of the program counter **21**, an instruction word **62** of the interrupt handler is read out to the instruction fetch section **20**, and the instruction execution section **30** sequentially executes processing toward the final address of the interrupt handler in accordance with the read-out instruction word.

At the final stage of the interrupt handler corresponding to the interrupt, the context is restored in step S10. When context restoration processing is ended, the flow advances to step S11. The instruction execution section **30** executes the break-interrupt return instruction to return the processor from instruction break interrupt processing to the previous application program processing.

More specifically, the instruction execution section **30** reads out the value of the previous state register **53** and writes the value in the present state register **54**. Thus, the processor transits from the supervisor state to the user state. The instruction execution section **30** also reads out, from the return address register **52**, the original instruction address to which the processor will return from the interrupt state, supplies the instruction address to the instruction fetch section **20** as a branch destination instruction address **65**, and sets the address value in the program counter **21**.

On the basis of the original instruction address set in the program counter **21**, the instruction fetch section **20** reads out the instruction word **62** for the normal operation from the memory **10**, temporarily holds the instruction word in the instruction register **22**, and then supplies the instruction word to the instruction execution section **30**. The instruction execution section **30** executes the remaining part of processing of the application corresponding to the normal operation.

As described above, when the breakpoint target instruction word is not a conditional instruction, and the condition of the conditional instruction is not satisfied, the flow immediately advances to step S10 without performing processing for the instruction break in step S9. In step S10, the context is restored. In step S11, return processing from the instruction break-interrupt is executed, and then, processing returns to the operation of the original application program.

As described above, in the 18th embodiment, whether the generation condition of the instruction break corresponding to the instruction break address and flag value is satisfied is determined by the determination sections 25_{-0} to 25_{-n}. In addition, whether the condition of the conditional instruction is satisfied is determined by software of the interrupt handler. Only when both conditions are satisfied, break-interrupt processing is actually executed.

Thus, like in the 10th embodiment, in debugging a program including a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the conditional instruction is satisfied, a break-interrupt occurs. When the condition of the conditional instruction is not satisfied, or the supplied instruction is an unconditional instruction, a break-interrupt can be inhibited.

In the 18th embodiment, processing corresponding to the scalar processor described in the 10th embodiment is implemented by the function of software. Processing corresponding to the VLIW type processor described in the 11th and 12th embodiments or the parallel processor described in the 13th and 14th embodiments can also be implemented by the function of software.

For example, to implement determination of the condition of a conditional instruction by the function of software,

	Document ID	U	Title	Current OR
1	JP 11282 679 A	☐	ARITHMETIC PROCESSOR	
2	JP 09091 118 A	☒	FLOATING POINT ARITHMETIC UNIT	
3	WO 99342 82 A1	☒	IMPROVED INSTRUCTION DISPATCH MECHANISM FOR A GUARDED VLIW ARCHITECTURE	
4	EP 60592 7 A1	☒	Improved very long instruction word processor architecture.	
5	NN910 282	☒	Computer Microcode Control With First Cycle Hardwired.	
6	US 66683 16 B	☒	Data processing device for very long instruction word architecture, includes output selector which selectively outputs integer and floating point results from integer and floating point computation units to write port of register file	
7	US 66292 32 B	☒	Electronic data processing system e.g. for very long instruction word processor, connects read, local write and remote write ports of register file copies to respective clusters of execution units	
8	US 64461 90 B	☒	Indirect very long instruction word data processor addresses registers using fields of instruction word corresponding to particular execution unit and contents of register file index registers	
9	US 63973 24 B	☒	Very long instruction word processor, has load and store units which share store read port of computer register file, for performing data-dependent load address and store address generation operations, respectively	
10	US 64153 76 B	☒	Issue grouping of instructions in very long instruction word processors into three groups using chaining bits to chain instructions appearing after end of last group	
11	US 62694 37 B	☒	Port pressure reduction of clustered processor, involves copying value from register and predicate file portion of cluster to respective file portion of another cluster by duplicator using inter-cluster move instruction	
12	WO 20007 9395 A	☒	Port priority function provider in word processor, resolves write priority conflict between instruction on single word basis to enable operation to complete on single word portion of register of double word width	
13	US 61638 37 A	☒	Writing of instruction results produced by two different instruction execution circuits to result destinations using separate write ports for the two circuits	
14	US 61548 28 A	☒	Parallel processing computer system has multiple processors to simultaneously execute operational instructions in instruction register based on state of cycle bit associated with operational instruction	
15	WO 20005 4144 A	☒	Register file indexing apparatus for very long instruction word processor, has controller associated with indexed part look ahead register allowing register to be addressed using fields of instruction word	
16	US 60761 54 A	☒	Very large instruction word processor	
17	US 58706 14 A	☒	Read crossbar simplification or elimination for very long instruction word processors - functional units are assigned to particular slots in the instruction issue register so the number of slots is less than number of units, and by associating a read port with each slot the read crossbar either simplified or eliminated	
18	US 57873 02 A	☒	VLIW processor using compressed instructions with instruction issue register - byte aligns variable length instructions, branch targets are uncompressed with format bits showing how many issue slots are used in following instruction, NOPS not stored in memory, compresses individual operations based on features	

that the instruction break generation condition is not satisfied. The determination sections 25_{-0} to 25_{-n} supply determination signals representing the determination results to the OR circuit **26**.

The OR circuit **26** performs OR operation to the determination signals supplied from all the determination sections 25_{-0} to 25_{-n}. When the instruction break generation condition is satisfied for at least one determination section, the OR circuit **26** detects a break-interrupt by an instruction breakpoint and notifies the interrupt control section **40** of the break-interrupt using an interrupt notification signal **67**.

The instruction execution section **30** executes processing such as calculation, branch, data load/store, or return from the interrupt state in accordance with the instruction of the instruction word **63** supplied from the instruction fetch section **20**. For example, if the supplied instruction is a calculation instruction, the instruction execution section **30** executes calculation on the basis of a data value **69** read out from a general-purpose register (GR) **55** in the register section **50**, which is designated by a register address **68**, and writes a data value **70** obtained by this calculation in the general-purpose register **55** designated by the register address **68**.

If the supplied instruction is a branch instruction, and the branch condition is satisfied, the instruction execution section **30** supplies the instruction address **64** of the branch destination to the instruction fetch section **120**. If the supplied instruction is a load instruction, the instruction execution section **30** obtains the effective address on the memory **10** from the data value **69** read out from the general-purpose register **55** designated by the register address **68**, reads out read data **71** from an area of the memory **10**, which corresponds to the effective address, and writes the read-out data in the general-purpose register **55** designated by the register address **68**.

When the supplied instruction is a store instruction, the instruction execution section **30** obtains the effective address on the memory **10** from the data value **69** read out from the general-purpose register **55** designated by the register address **68**, and writes the data value **69** read out from the general-purpose register **55** designated by the register address **68** in an area of the memory **10**, which corresponds to the effective address.

When the supplied instruction is an instruction for return from the interrupt state, the instruction execution section **30** executes the return operation from the interrupt state. More specifically, on the basis of operation information before the interrupt, which is held in each register of the register section **50**, processing of restoring the operation information before the interrupt is executed.

In this case, the value of a previous state register (EPSR) **53** is written in a present state register (PSR) **54**, and simultaneously, the value of a return address register (EPCR) **52** is read out, and the read-out return address is supplied to the instruction fetch section **20** as the branch destination address **65**.

When an interrupt due to an error such as division by zero or data overflow is detected in executing a calculation instruction by the instruction execution section **30**, the instruction execution section **30** notifies the interrupt control section **40** of the interrupt using an interrupt notification signal **74**. The interrupt due to an error is called a normal interrupt and discriminated from a break-interrupt by an instruction break or software break.

When receiving the interrupt notification signal **67** for a break-interrupt from the instruction fetch section **20**, or the interrupt notification signal **74** for a normal interrupt from the instruction execution section **30**, the interrupt control section **40** controls the instruction fetch section **20** and the register section **50** to execute the shift operation to the interrupt state.

More specifically, when receiving the interrupt notification signal **67** or **74**, the interrupt control section **40** reads out an instruction address **73** at the time of interrupt from the program counter **21** of the instruction fetch section **20** and writes the instruction address **73** in the return address register **52** of the register section **50**. The start address **66** of the interrupt handler corresponding to the interrupt that has occurred is supplied to the instruction fetch section **20** and set in the program counter **21**. The interrupt control section **40** also writes the processor state before the interrupt in the previous state register **53** and writes, in the present state register **54**, the processor state that has transited in accordance with the interrupt.

The register section **50** has a condition register **51** in addition to the above-described return address register **52**, previous state register **53**, present state register **54**, and general-purpose register **55**. This condition register **51** holds a condition code that is referred to when the instruction execution section **30** executes a conditional instruction, which is supplied from the instruction fetch section **20**. A conditional instruction means an instruction for which it is determined first whether a designated condition is satisfied, and only when the condition is satisfied, designated processing such as data transfer or calculation is executed.

The return address register **52** holds the original instruction address (the value **73** of the program counter **21** at the time of interrupt) to which the processor will return from the interrupt state. The previous state register **53** holds the processor state (normal user state without any interrupt or supervisor state that has transited in accordance with the interrupt) before the interrupt. The present state register **54** holds the present processor state.

The state transition of the processor will be briefly described.

When the processor is processing a normal application program, the processor state is the user state. When an interrupt occurs in this user state, the processor transits to the supervisor state. When an interrupt return instruction is executed in this supervisor state, the processor returns to the user state.

On the other hand, if another interrupt occurs before the interrupt return instruction is executed in the supervisor state, the processor transits from the current supervisor state to the next supervisor state. When the interrupt return instruction is executed in this new supervisor state, the processor returns to the previous supervisor state.

The operation of the processor shown in FIG. **3** will be described next. When the processor is in the user state, the instruction fetch section **20** reads out the instruction word **62** from the memory **10** on the basis of the instruction address **61** indicated by the program counter **21**, and writes/holds the read-out instruction word in the instruction register **22**. The instruction fetch section **20** also supplies the instruction word **63** held in the instruction register **22** to the instruction execution section **30**.

The instruction execution 'section **30** decodes the instruction word **63** supplied from the instruction fetch section **20** and executes processing of the supplied instruction in accordance with the decoding result. If no break-interrupt or normal interrupt occurs in this user state, the processor repeats the above operation.

However, when the instruction fetch section **20** detects a break-interrupt, the instruction fetch section **20** notifies the

	Document ID	U	Title	Current OR
19	EP 605927 A	☐	Very long word processor architecture - distributes instructions to selected functional units which access register file via switching matrices that reduce number of parallel ports	

interrupt control section **40** of the break-interrupt using the interrupt notification signal **67**. When the instruction execution section **30** detects a normal interrupt, the instruction execution section **30** notifies the interrupt control section **40** of the normal interrupt using the interrupt notification signal **74**.

When receiving the interrupt notification from the instruction fetch section **20** or instruction execution section **30**, the interrupt control section **40** controls the instruction fetch section **20** and register section **50** to perform processing as follows.

First, the interrupt control section **40** reads out the currently indicated instruction address **73** from the program counter **21** and writes the read-out instruction address **73** in the return address register **52**.

The interrupt control section **40** also reads out the processor state (user state) before the interrupt from the present state register **54**, writes the read-out processor state in the previous state register **53** and, also writes, in the present state register **54**, the processor state that has transited in accordance with the interrupt. Thus, the processor transits from the user state to the supervisor state. The interrupt control section **40** also supplies the start address **66** of the interrupt handler corresponding to the interrupt to the instruction fetch section **20** and sets the address value in the program counter **21**.

The processor which has transited to the supervisor state reads out the instruction word **62** of the interrupt handler to the instruction fetch section **20** in accordance with the start address **66** of the interrupt handler, which is set in the program counter **21**, temporarily holds the read-out instruction word in the instruction register **22**, and then supplies the instruction word to the instruction execution section **30**.

The instruction execution section **30** decodes the supplied instruction word **63** and executes processing in accordance with the decoding result. At this time, the instruction execution section **30** repeats the operation of executing the instruction word **63** sequentially supplied toward the end address of the interrupt handler. When processing of the interrupt handler corresponding to the interrupt is ended, the processor executes the interrupt return instruction.

When receiving the interrupt return instruction, the instruction execution section **30** reads out the value of the previous state register **53** and writes the value in the present state register **54**. Thus, the processor transits from the supervisor state to the user state. The instruction execution section **30** also reads out the original instruction address to which the processor will return from the interrupt state from the return address register **52**, supplies the instruction address to the instruction fetch section **20** as the branch destination address **65**, and sets the instruction address i n the program counter **21**.

On the basis of the original instruction address **61** set in the program counter **21**, the instruction fetch section **20** reads out the instruction word **62** for the normal operation from the memory **10**, temporarily holds the instruct ion word **62** in the instruction register **22**, and then supplies the instruction word to the instruction execution section **30**. The instruction execution section **30** executes processing of the remaining part of the application program corresponding to the normal operation.

To set execution of a break-interrupt by an instruction break, for an entry of interest in entries #0 to #n corresponding to the breakpoint registers 24_{-0} to 24_{-n} in the instruction break detection section **23**, the target address of a breakpoint is written in the address register **24***a*, and the value "1" representing the instruction break operation valid state is written in the flag register **24***b*.

To cancel execution of the break-interrupt by an instruction break, for an entry of interest in the entries #0 to #n corresponding to the breakpoint registers 24_{-0} to 24_{-n} in the instruction break detection section **23**, the value "0" representing the instruction break operation invalid state is written in the flag register **24***b*.

FIG. **4** is a block diagram showing a conventional construction for implementing the software break scheme by a breakpoint instruction.

In FIG. **4**, the same reference numerals as in FIG. **3** denote the same functional parts as in FIG. **3**, respectively, and a detailed description thereof will be omitted.

Referring to FIG. **4**, when detecting a normal interrupt due to an instruction address conversion error or the like, an instruction fetch section **20** notifies an interrupt control section **40** of the normal interrupt using an interrupt notification signal **81**.

When a breakpoint instruction is supplied in executing an instruction supplied from the instruction fetch section **20** by an instruction execution section **30**, the instruction execution section **30** notifies the interrupt control section **40** of the software break-interrupt using an interrupt notification signal **82**.

FIG. **5** is a representation showing the construction of a breakpoint table held by the instruction execution section **30** in the software break scheme.

Referring to FIG. **5**, a column "VALID" represents whether the breakpoint operation is valid. A value "1" means a breakpoint operation valid state while a value "0" means a breakpoint operation invalid state.

A column "ADDRESS" holds the target address of a breakpoint at which execution is to be stopped. A column "INSTRUCTION" holds a breakpoint target instruction word. The breakpoint target instruction word is replaced with a breakpoint instruction for generating a break-interrupt. For this reason, in canceling the breakpoint operation, restoration is done using the data of the breakpoint target instruction word held in the column "INSTRUCTION".

To set execution of a break-interrupt by a breakpoint instruction, for an entry of interest in entries #0 to #n of the breakpoint table shown in FIG. **5**, an instruction address representing a breakpoint is written in the column "ADDRESS", a breakpoint target instruction word is written in the column "INSTRUCTION", and the value "1" representing the breakpoint operation valid state is written in the column "VALID". In addition, the breakpoint target instruction word is replaced with a breakpoint instruction for generating a break-interrupt.

Contrastingly, to cancel execution of the break-interrupt by a breakpoint instruction, for an entry of interest in the entries #0 to #n of the breakpoint table shown in FIG. **5**, the breakpoint target instruction word written in the column "INSTRUCTION" is read out and replaced with a breakpoint instruction. In addition, for the entry of interest, the value "0" representing the breakpoint operation invalid state is written in the column "VALID".

Recent processors are required to make the branch instruction use frequency low in order to improve the processing performance. For this purpose, techniques including a conditional instruction or predicated execution have been proposed ("HPL PlayDoh Architecture Specification ver. 1.0, "Vinod Kathail Etc HPL 93-80 February 1994, "Incorporating Guarded Execution into Existing Instruction Set" D. N. Pnevmatikatos PDH Paper Wisconsin Univ. 1996, and "The Benefit of Predicated Execution for Software Pipelining" N. J. Warter etc. IIICSS-26 Conference Proceedings January 1993 Vol. 1, pp. 497–606).

	Document ID	U	Title	Current OR
1	JP 20031 34000 A	☐	DATA TRANSCEIVING EQUIPMENT AND COMMUNICATION APPARATUS	
2	JP 20030 37572 A	☒	SCHEDULING SYSTEM	
3	JP 20020 99930 A	☒	SEAT RESERVATION TICKET ISSUING MACHINE AND TICKET-ISSUING MACHINE	
4	JP 20010 53829 A	☒	DEVICE AND METHOD FOR CONTROLLING COMMUNICATION	
5	JP 20001 49090 A	☒	AUTOMATIC CHANGE DISPENSER	
6	JP 11268 376 A	☒	PORTABLE PRINTER	
7	JP 08292 824 A	☒	CIRCUIT AND METHOD FOR COMPUTER RESET CONTROL	
8	JP 08221 334 A	☒	DEVICE AND METHOD FOR SETTING DEVICE ADDRESS	
9	JP 08171 669 A	☒	CHANGE CERTIFICATION TICKET ISSUING DEVICE	
10	JP 06202 896 A	☒	SYSTEM STORAGE DEVICE	
11	JP 05266 280 A	☒	TICKET ISSUING MACHINE PROVIDED WITH REPAYMENT FUNCTION	
12	JP 05225 361 A	☒	REGISTER REWRITING SYSTEM	
13	JP 05143 198 A	☒	INFORMATION PROCESSOR	
14	JP 05016 746 A	☒	COMMUNICATION DEVICE FOR MOBILE BODY	
15	JP 03189 845 A	☒	HIERARCHICAL MEMORY SYSTEM AND CACHE MEMORY SUBSYSTEM	
16	JP 01204 150 A	☒	INFORMATION PROCESSOR	
17	JP 60163 139 A	☒	MICROCOMPUTER	
18	JP 57203 156 A	☒	COMPUTER FOR CONTROL	
19	JP 57116 410 A	☒	KARMAN TYPE EQUALIZER	
20	WO 99611 99 A1	☒	METHOD AND APPARATUS FOR CHANGING OPERATING MODULES ON A COORDINATE POSITIONING MACHINE	
21	GB 22666 06 A	☒	A microprocessor with an external command mode for diagnosis and debugging	

However, in the conventional interrupt scheme using an instruction break, as shown in FIG. 3, a break-interrupt always occurs when the instruction break generation condition set in each of the breakpoint registers 24_0 to 24_n of the instruction break detection section 23 is satisfied. For this reason, in debugging a program containing a conditional instruction, execution of the program is interrupted when the instruction break generation condition is satisfied while the condition of the conditional instruction is not satisfied.

In the conventional interrupt scheme using a breakpoint instruction, as shown in FIG. 4, a break-interrupt always occurs when a breakpoint instruction that has been replaced in advance is supplied. For this reason, in debugging a program containing a conditional instruction, execution of the program is interrupted when the breakpoint instruction is supplied while the condition of the conditional instruction is not satisfied.

SUMMARY OF THE INVENTION

It is the first object of the present invention to generate a break-interrupt even within the interrupt inhibition period.

It is the second object of the present invention to control generation of a break-interrupt in debugging a program containing a conditional instruction depending on whether the condition of the conditional instruction is satisfied.

In order to achieve the first object, according to the present invention, there is provided an interrupt control apparatus having a function of normal interrupt and a function of break-interrupt, comprising a first information holding section for holding, at the time of a normal interrupt, operation information of a processor before the interrupt, a second information holding section for holding, at the time of a break-interrupt operation information of the processor before the break-interrupt, a return operation specifying section for specifying whether or not a return operation from a normal interrupt state or a return operation from a break-interrupt state is to be performed in returning from an interrupt operation, and an interrupt return section for re-setting operation information held in the first information holding section or operation information held in the second information holding section in accordance with operation contents specified by the return operation specifying section, and thereby returning a state of the processor from an interrupt operation state to a state before the interrupt.

According to the present invention, there is also provided an interrupt control method, comprising the steps of, when a normal interrupt occurs, holding operation information of a processor before the normal interrupt in a first information holding section, when a break-interrupt occurs, holding operation information of the processor before the break-interrupt in a second information holding section different from the first information holding section and setting a flag for showing whether or not the break-interrupt state is set, to the break-interrupt state, and in returning the processor from the interrupt state to a state before the interrupt, selecting and restoring one of operation information in the first information holding section and operation information in the second information holding section in accordance with a value of the flag.

According to the present invention having the above construction, when a break-interrupt occurs, processor operation information before the break-interrupt is saved in a holding section dedicated to the break-interrupt independently of that for a normal interrupt. In returning from the break-interrupt, the processor operation information before the break-interrupt is restored from the holding section dedicated to the break-interrupt. Thus, the operation information before a normal interrupt, which is required for returning from the normal interrupt state is not overwritten by the operation information saved at the time of break-interrupt. Hence, a break-interrupt can occur even within an interrupt inhibition period by a normal interrupt.

More specifically, according to the present invention, the processor operation information at the time of break-interrupt can be held even within interrupt inhibition periods immediately after the normal interrupt operation and immediately before interrupt return, so a break-interrupt can occur even within the interrupt inhibition period. Hence, when an interrupt handler is generated as part of an application, the interrupt handler can be completely debugged.

For example, only an instruction address is held as the processor operation information before a break-interrupt, which is to be held when a break-interrupt occurs, the return operation from the break-interrupt state can be executed with minimum necessary operation information.

Additionally, according to the present invention, in returning from an interrupt state, it can be specified whether the operation is a return operation from a break-interrupt state or normal interrupt state by referring to a flag representing whether the break-interrupt state is set. For this reason, even for a break-interrupt that has occurred within the interrupt inhibition period by a normal interrupt, the processor operation information before the break-interrupt can be accurately restored, and a break-interrupt can occur even within the interrupt inhibition period by the normal interrupt.

According to another characteristic feature of the present invention, in returning from the interrupt state, it can be specified in accordance with the contents of an interrupt return instruction whether the operation is a return operation from a break-interrupt state or normal interrupt state. For this reason, even for a break-interrupt that has occurred in the normal interrupt state, the processor operation information before the break-interrupt can be accurately restored without preparing the flag representing whether the break-interrupt state is set. Hence, a break-interrupt can be generated using a small hardware resource.

In order to achieve the second object, according to the present invention, there is provided an interrupt control apparatus applied to a data processing system having a function of executing a conditional instruction, comprising a break detection section for detecting a breakpoint set at an arbitrary position of an instruction sequence, a condition determination section for determining whether or not a condition of the conditional instruction is satisfied, and a control section for controlling a break-interrupt on the basis of a breakpoint detection result from the break detection section and a determination result from the condition determination section.

More specifically, the apparatus comprises an instruction break detection section for detecting an instruction break in accordance with whether or not an instruction corresponding to an instruction address representing a breakpoint, which is set in a register, is read out, and outputting a detection signal representing a detection result, a condition determination section for determining whether or not a condition of the read-out conditional instruction is satisfied and outputting a determination signal representing a determination result, and a logical operation section for performing AND operation to the detection signal output from the instruction break detection section and the determination signal output from the condition determination section and sending a break-interrupt notification in accordance with an AND operation result.

	Document ID	U	Title	Current OR
22	US 6609165 B	☒	Extended link service command transmitting apparatus used in computing environment, has primary port that receives notification regarding change in link service command from secondary port of fiber link having no fabric controller	
23	US 6272624 B	☒	Branch outcome prediction system includes register providing pattern of program control flow, that is modified based on summary of control flow activity of group of instructions fetched in given slot	
24	JP 08313167 A	☒	Integrated system of gas treatment with reusable inert gas - is connected to gas take-over port of hot static water pressuriser, providing substitution for pressure medium gas, etc.	
25	EP 588084 A	☒	Lap-top or notebook portable computer with dedicated register group and peripheral controller bus - has dedicated register group between system bus and peripheral controllers, for temporarily storing control data, so that CPU reads and writes to group through system bus, and controllers read and write to group via controller bus	
26	GB 2266606 A	☒	Computer circuit with external command mode for diagnosis and debugging - responds to series of stored instructions from computer memory and operates in external command mode responsive to externally generated instruction and externally generated command	
27	RD 313124 A	☒	Transmission oil life diagnostic method for motor vehicle - using continuous recording of oil stress dependent on operating temperature to indicate remaining oil life	
28	GB 2143361 A	☒	Industrial plant simulator - has digital control and programmer for affecting plant operation and has indicators mounted on console	
29	CA 1145021 A	☐	Port event timing analysis system - uses register to store command timing data and timer to generate time interval signals for altering supervisory data	

According to another aspect of the present invention, there is provided an interrupt control apparatus applied to a data processing system having a function of executing a conditional instruction, comprising an instruction break detection section for detecting an instruction break in accordance with whether or not an instruction corresponding to an instruction address representing a breakpoint, which is set in a register, is read out, and sending a break-interrupt notification in accordance with a detection result, and a control section for, in an interrupt handler activated in accordance with the break-interrupt notification supplied from the instruction break detection section, determining whether or not a condition of the conditional instruction is satisfied and controlling break-interrupt processing in accordance with a determination result.

According to still another aspect of the present invention, there is provided an interrupt control apparatus applied to a data processing system having a function of executing a conditional instruction, comprising a software break detection section for detecting a software break in accordance with whether a breakpoint instruction replaced at an arbitrary position of an instruction sequence is executed and sending a break-interrupt notification in accordance with a detection result, and a control section for, in an interrupt handler activated in accordance with the break-interrupt notification supplied from the software break detection section, determining whether or not a condition of the conditional instruction is satisfied and controlling break-interrupt processing in accordance with a determination result.

According to the present invention, there is also provided an interrupt control method of controlling a break-interrupt in a data processing system having a function of executing a conditional instruction, comprising the steps of detecting a breakpoint set at an arbitrary position of an instruction sequence, determining whether or not a condition of the conditional instruction is satisfied, and controlling the break-interrupt on the basis of a detection result of the breakpoint and a determination result of the conditional instruction.

According to the present invention having the above construction, a break-interrupt can be controlled not only on the basis of the detection result of a breakpoint such a an instruction break or software break but also, when the supplied instruction is a conditional instruction, on the basis of the determination result of the condition. Hence, in debugging a program including a conditional instruction, when the condition of the conditional instruction is satisfied, program execution is interrupted. When the condition of the conditional instruction is not satisfied, an interrupt of the program execution can be inhibited.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a representation showing an example of state transition of a processor in a conventional interrupt control apparatus;

FIG. **2** is a representation showing another example of state transition of the processor in the conventional interrupt control apparatus;

FIG. **3** is a block diagram showing the construction of a processor for implementing the conventional instruction break scheme;

FIG. **4** is a block diagram showing the construction of a processor for implementing the conventional software break scheme;

FIG. **5** is a representation showing the construction of a breakpoint table used in the software break scheme;

FIG. **6** is a block diagram showing the construction of an interrupt control apparatus according to the first embodiment of the present invention;

FIG. **7** is a representation showing an example of state transition of a processor in the interrupt control apparatus according to the first embodiment;

FIG. **8** is a representation showing another example of state transition of the processor in the interrupt control apparatus according to the first embodiment;

FIG. **9** is a block diagram showing the construction of an interrupt control apparatus according to the second embodiment of the present invention;

FIG. **10** is a block diagram showing the construction of an interrupt control apparatus according to the third embodiment of the present invention;

FIG. **11** is a block diagram showing the construction of an interrupt control apparatus according to the fourth embodiment of the present invention;

FIG. **12** is a block diagram showing the construction of an interrupt control apparatus according to the fifth embodiment of the present invention;

FIG. **13** is a representation showing the instruction form of an interrupt return instruction used in the fifth embodiment and also the following sixth to ninth embodiments of the present invention;

FIG. **14** is a block diagram showing the construction of an interrupt control apparatus according to the sixth embodiment of the present invention;

FIG. **15** is a block diagram showing the construction of an interrupt control apparatus according to the seventh embodiment of the present invention;

FIG. **16** is a block diagram showing the construction of an interrupt control apparatus according to the eighth embodiment of the present invention;

FIG. **17** is a block diagram showing the construction of an interrupt control apparatus according to the ninth embodiment of the present invention;

FIG. **18** is a block diagram showing the construction of a data processing system (processor) according to the 10th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **19** is a block diagram showing the construction of a determination section according to the 10th embodiment;

FIG. **20** is a block diagram showing the construction of a data processing system (processor) according to the 11th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **21** is a block diagram showing the construction of a determination section according to the 11th embodiment;

FIG. **22** is a block diagram showing the construction of a data processing system (processor) according to the 12th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **23** is a block diagram showing the construction of a determination section according to the 12th embodiment;

FIG. **24** is a block diagram showing the construction of a data processing system (processor) according to the 13th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **25** is a block diagram showing the construction of a determination section according to the 13th embodiment;

FIG. **26** is a block diagram showing the construction of a data processing system (processor) according to the 14th

	Document ID	U	Title	Current OR
1	US 20040039896 A1	☐	Methods and apparatus for meta-architecture defined programmable instruction fetch functions supporting assembled variable length instruction processors	712/205
2	US 20040030860 A1	☒	Conditional execution control head in a vliw processor	712/24
3	US 20040015599 A1	☒	Network processor architecture	709/232
4	US 20040003212 A1	☒	Data processor	712/229
5	US 20030226003 A1	☒	Information processor having delayed branch function	712/238
6	US 20030212524 A1	☒	Test access circuit and method of accessing embedded test controllers in integrated circuit modules	702/120
7	US 20030200539 A1	☒	Function unit based finite state automata data structure, transitions and methods for making the same	717/161
8	US 20030196197 A1	☒	Methods and systems for integrated scheduling and resource management for a compiler	717/161
9	US 20030196040 A1	☒	Data cache system	711/128
10	US 20030177340 A1	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/219
11	US 20030154358 A1	☒	Apparatus and method for dispatching very long instruction word having variable length	712/24
12	US 20030145116 A1	☒	System for communication with a storage area network	709/249
13	US 20030126404 A1	☒	Data processing system, array-type processor, data processor, and information storage medium	712/15
14	US 20030093655 A1	☒	Multithread embedded processor with input/output capability	712/228
15	US 20030074533 A1	☒	Instruction pair detection and pseudo ports for cache array	711/125
16	US 20030074532 A1	☒	Instruction pair detection and pseudo ports for cache array	711/123
17	US 20030005231 A1	☒	Hardware emulation of parallel ATA drives with serial ATA interface	711/131

embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **27** is a block diagram showing the construction of a determination section according to the 14th embodiment;

FIG. **28** is a block diagram showing the construction of a data processing system (processor) according to the 15th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **29** is a block diagram showing the first construction of a determination section according to the 15th embodiment;

FIG. **30** is a block diagram showing the second construction of the determination section according to the 15th embodiment;

FIG. **31** is a block diagram showing the third construction of the determination section according to the 15th embodiment;

FIG. **32** is a block diagram showing the fourth construction of the determination section according to the 15th embodiment;

FIG. **33** is a block diagram showing the fifth construction of the determination section according to the 15th embodiment;

FIG. **34** is a block diagram showing the construction of a data processing system (processor) according to the 16th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **35** is a block diagram showing the first construction of a determination section according to the 16th embodiment;

FIG. **36** is a block diagram showing the second construction of the determination section according to the 16th embodiment;

FIG. **37** is a block diagram showing the third construction of the determination section according to the 16th embodiment;

FIG. **38** is a block diagram showing the fourth construction of the determination section according to the 16th embodiment;

FIG. **39** is a block diagram showing the fifth construction of the determination section according to the 16th embodiment;

FIG. **40** is a block diagram showing the construction of a data processing system (processor) according to the 17th embodiment of the present invention for implementing an instruction break scheme by a hardware mechanism;

FIG. **41** is a block diagram showing the first construction of a determination section according to the 17th embodiment;

FIG. **42** is a block diagram showing the second construction of the determination section according to the 17th embodiment;

FIG. **43** is a block diagram showing the third construction of the determination section according to the 17th embodiment;

FIG. **44** is a block diagram showing the fourth construction of the determination section according to the 17th embodiment;

FIG. **45** is a block diagram showing the fifth construction of the determination section according to the 17th embodiment;

FIG. **46** is a block diagram showing the construction of a determination section according to the 18th to 21st embodiments of the present invention;

FIG. **47** is a flow chart showing the first example of processing by an instruction break-interrupt handler according to the 18th embodiment;

FIG. **48** is a representation showing the first example of construction of a breakpoint table used in the 18th embodiment;

FIG. **49** is a flow chart showing the second example of processing by the instruction break-interrupt handler according to the 18th embodiment;

FIG. **50** is a flow chart showing the third example of processing by the instruction break-interrupt handler according to the 18th embodiment;

FIG. **51** is a representation showing the second example of construction of the breakpoint table used in the 18th embodiment;

FIG. **52** is a flow chart showing the fourth example of processing by the instruction break-interrupt handler according to the 18th embodiment;

FIG. **53** is a flow chart showing the fifth example of processing by the instruction break-interrupt handler according to the 18th embodiment;

FIG. **54** is a flow chart showing processing by an instruction break-interrupt handler according to the 19th embodiment;

FIG. **55** is a representation showing the first example of construction of a breakpoint table used in the 19th embodiment;

FIG. **56** is a representation showing the second example of construction of the breakpoint table used in the 19th embodiment;

FIG. **57** is a flow chart showing processing by an instruction break-interrupt handler according to the 20th embodiment;

FIG. **58** is a flow chart showing processing by an instruction break-interrupt handler according to the 21st embodiment;

FIG. **59** is a representation showing the first example of construction of a breakpoint table used in the 22nd embodiment of the present invention;

FIG. **60** is a representation showing the second example of construction of the breakpoint table used in the 22nd embodiment; and

FIG. **61** is a representation showing the third example of construction of the breakpoint table used in the 22nd embodiment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Hereinafter, embodiments of the present invention will be described with reference to drawings.

First Embodiment

FIG. **6** is a block diagram showing the construction of an interrupt control apparatus according to the first embodiment of the present invention.

Referring to FIG. **6**, reference numeral **410** denotes a memory which stores programs including an application and interrupt handler; **420** denotes an instruction fetch section; **430** denotes an instruction execution section; **440** denotes an interrupt control section; and **450** denotes a register section.

The instruction fetch section **420** comprises an instruction fetch controller **421**, a program counter **422**, and an instruction word register **423**. The instruction fetch controller **421**

	Document ID	U	Title	Current OR
18	US 20020144092 A1	☒	Handling of loops in processors	712/217
19	US 20020144088 A1	☒	Apparatus and method for issue grouping of instructions in a VLIW processor	712/210
20	US 20020133784 A1	☒	Automatic design of VLIW processors	716/1
21	US 20020120914 A1	☒	Automatic design of VLIW processors	716/17
22	US 20020108026 A1	☒	Data processing apparatus with register file bypass	712/218
23	US 20020099928 A1	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/216
24	US 20020087835 A1	☒	Method and apparatus for improving dispersal performance in a processor through the use of no-op ports	712/215
25	US 20020052926 A1	☒	Thread suspension system and method using trapping instructions	709/217
26	US 20020042908 A1	☒	Compiler parallelizing schedule method	717/149
27	US 20020023203 A1	☒	Memory access debug facility	712/227
28	US 20020026399 A1	☒	Methods and apparatus for loading a very long instruction word memory	710/22
29	US 20010052053 A1	☒	Stream processing unit for a multi-streaming processor	711/138
30	US 20010042187 A1	☒	VARIABLE ISSUE-WIDTH VLIW PROCESSOR	712/2
31	US 20010032304 A1	☒	Processor for making more efficient use of idling components and program conversion apparatus for the same	712/24
32	US 20010023479 A1	☒	Information processing unit, and exception processing method for specific application-purpose operation instruction	712/209
33	US 20010016901 A1	☒	Communicating instruction results in processors and compiling methods for processors	712/217
34	US 20010016900 A1	☒	Dynamic allocation of resources in multiple microprocessor pipelines	712/215

reads out an instruction word **472** from the memory **410** on the basis of an instruction address **471** indicated by the program counter **422** and writes/holds the read-out instruction word in the instruction register **423**. The instruction fetch section **420** also supplies an instruction word **473** held in the instruction register **423** to the instruction execution section **430**.

When an instruction address **474** of a branch destination or an instruction address **475** for return from the interrupt state is supplied from the instruction execution section **430**, or when an instruction address **476** of a normal interrupt handler or an instruction address **477** of a break-interrupt handler is supplied from the interrupt control section **440**, the instruction fetch section **420** writes the supplied instruction address in the program counter **422**. Otherwise, the value of the program counter **21** is incremented by one to read out the next instruction word sequentially.

When a break-interrupt by an instruction breakpoint or step execution is detected in reading out the instruction word **472** from the memory **410**, the instruction fetch controller **421** notifies the interrupt control section **440** of the break-interrupt using a break-interrupt notification signal **478**. When a normal interrupt due to an instruction address conversion error or the like is detected, the instruction fetch controller **421** notifies the interrupt control section **440** of the normal interrupt using a normal interrupt notification signal **479**.

The instruction execution section **430** comprises an instruction word decoder **431**, an instruction execution controller **432**, a breakpoint controller **433**, and an interrupt return controller **434**. The instruction word decoder **431** decodes the instruction word **473** supplied from the instruction fetch section **420**.

If the supplied instruction word **473** is an instruction word for generating a break-interrupt by a software breakpoint, a break-interrupt generation instruction is supplied to the breakpoint controller **433**. If the supplied instruction word **473** is an instruction word for returning the processor from the interrupt state, an interrupt return instruction is supplied to the interrupt return controller **434**. If the supplied instruction word **473** is an instruction word of another type, the decoded instruction is supplied to the instruction execution controller **432**.

The instruction execution controller **432** executes processing such as calculation, branch, or data load/store in accordance with the instruction supplied from the instruction word decoder **431**. For example, if the supplied instruction is a calculation instruction, the instruction execution controller **432** executes calculation on the basis of a data value **481** read out from a general-purpose register (GR) **451** in the register section **450**, which is designated by a register address **480**, and writes a data value **482** obtained by this calculation in the general-purpose register **451** designated by the register address **480**.

If the supplied instruction is a branch instruction, and the branch condition is satisfied, the instruction execution controller **432** supplies the instruction address **474** of the branch destination to the instruction fetch section **420**. If the supplied instruction is a load instruction or a store instruction, the instruction execution controller **432** obtains the effective address on the memory **410** from the data value **481** read out from the general-purpose register **451** designated by the register address **480**, and reads out read data **483** or writes write data **484** from/in an area of the memory **410**, which corresponds to the effective address.

When detecting a break-interrupt by a data breakpoint, the instruction execution controller **432** notifies the interrupt control section **440** of the break-interrupt using a break-interrupt notification signal **485**. When a normal interrupt due to an error such as division by zero or data overflow is detected in executing a calculation instruction, the instruction execution controller **432** notifies the interrupt control section **440** of the normal interrupt using a normal interrupt notification signal **486**.

In accordance with the break-interrupt generation instruction supplied from the instruction word decoder **431**, the breakpoint controller **433** notifies the interrupt control section **440** of the break-interrupt using a break-interrupt notification signal **487**.

The interrupt return controller **434** executes a return operation from the interrupt state in accordance with the interrupt return instruction supplied from the instruction word decoder **431**. At this time, the interrupt return controller **434** specifies, on the basis of pieces of operation information before the interrupt, which are held by registers in the register section **450**, whether the operation is a return operation from a normal interrupt state or break-interrupt state, and restores the operation information before the interrupt.

The interrupt control section **440** comprises a normal interrupt controller **441** and break-interrupt controller **442**. When receiving the normal interrupt notification signal **479** from the instruction fetch section **420** or the normal interrupt notification signal **486** from the instruction execution section **430**, the normal interrupt controller **441** controls the instruction fetch section **420** and register section **450** to execute a shift operation to the normal interrupt state.

When receiving the normal interrupt notification signal **479** or **486**, the normal interrupt controller **441** reads out an instruction address **488** at the time of normal interrupt from the instruction fetch section **420**, supplies a start address **476** of the normal interrupt handler to the instruction fetch section **420**, and sets the address in the program counter **422**. The normal interrupt controller **441** also writes received pieces of information on the normal interrupt, e.g., pieces of information on the instruction address at the time of normal interrupt in registers in the register section **450**.

When receiving the break-interrupt notification signal **478** from the instruction fetch section **420** or one of the break-interrupt notification signals **485** and **487** from the instruction execution section **430**, the break-interrupt controller **442** controls the instruction fetch section **420** and register section **450** to execute a shift operation to the break-interrupt state.

When receiving the break-interrupt notification signal **478**, **485**, or **487**, the break-interrupt controller **442** loads an instruction address **489** at the time of break-interrupt from the instruction fetch section **420**, supplies the start address **477** of the break-interrupt handler to the instruction fetch section **420**, and sets the address in the program counter **422**. The break-interrupt controller **442** also writes received pieces of information on the break-interrupt, e.g., pieces of information on the instruction address at the time of break-interrupt in registers in the register section **450**.

The register section **450** includes the general-purpose register **451** for holding data to be used for calculation or the like by the instruction execution section **430**, and registers **452** to **457** for holding data to be used for interrupt control (to be described below). The registers **452** to **457** for interrupt control will be described below.

The normal return address register (EPCR) **452** holds the original instruction address (the value **488** of the program counter **422** at the time of normal interrupt) to which the

	Document ID	U	Title	Current OR
35	US 20010014940 A1	☒	Dynamic allocation of resources in multiple microprocessor pipelines	712/218
36	US 20010014939 A1	☒	Dynamic allocation of resources in multiple microprocessor pipelines	712/218
37	US 20010011342 A1	☒	Methods and apparatus for dynamic instruction controlled reconfigurable register file with extended precision	712/16
38	US 20010010073 A1	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/218
39	US 6691222 B2	☒	Non-stalling circular counterflow pipeline processor with recorder buffer	712/219
40	US 6684320 B2	☒	Apparatus and method for issue grouping of instructions in a VLIW processor	712/24
41	US 6668316 B1	☒	Method and apparatus for conflict-free execution of integer and floating-point operations with a common register file	712/221
42	US 6654870 B1	☒	Methods and apparatus for establishing port priority functions in a VLIW processor	712/24
43	US 6654869 B1	☒	Assigning a group tag to an instruction group wherein the group tag is recorded in the completion table along with a single instruction address for the group to facilitate in exception handling	712/24
44	US 6654834 B1	☒	Method and apparatus for data transfer employing closed loop of memory nodes	710/107
45	US 6651247 B1	☒	Method, apparatus, and product for optimizing compiler with rotating register assignment to modulo scheduled code in SSA form	717/161
46	US 6651222 B2	☒	Automatic design of VLIW processors	716/1
47	US 6629232 B1	☒	Copied register files for data processors having many execution units	712/29
48	US 6622234 B1	☒	Methods and apparatus for initiating and resynchronizing multi-cycle SIMD instructions	712/22
49	US 6598063 B1	☒	Fast calculation of (A/B)K by a parallel floating-point processor	708/606
50	US 6594713 B1	☒	Hub interface unit and application unit interfaces for expanded direct memory access processor	710/31
51	US 6581187 B2	☒	Automatic design of VLIW processors	716/1
52	US 6574724 B1	☒	Microprocessor with non-aligned scaled and unscaled addressing	711/220
53	US 6571329 B1	☒	Detection of overwrite modification by preceding instruction possibility of fetched instruction code using fetched instructions counter and store target address	712/205
54	US 6560674 B1	☒	Data cache system	711/118
55	US 6553485 B2	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/219

processor will return from the normal interrupt state. The normal previous state register (EPSR) **453** holds the processor state before the normal interrupt (normal user state or supervisor state). The normal factor register (ECR) **454** holds the factor of a normal interrupt. The values of these registers **452** to **454** are set at the time of normal interrupt. The normal return address register **452**, normal previous state register **453**, and normal factor register **454** constitute the first information holding section of the present invention.

The break return address register (BEPCR) **455** holds the original instruction address (the value **489** of the program counter **422** at the time of break-interrupt) to which the processor will return from the break-interrupt state. The value of this register is set at the time of break-interrupt. This break return address register **455** constitutes the second information holding section of the present invention. The instruction address set in the normal return address register **452** or break return address register **455** is supplied to the instruction fetch section **420** as the return address **475** in returning from the interrupt operation, and the address value is set in the program counter **422**.

The flag register (BE) **456** represents whether a break-interrupt state is set. The value "0" indicates a non-break-interrupt state, and the value "1" indicates a break-interrupt state. The initial value of the flag register **456** is "0". When a break-interrupt occurs, the value transits from "0" to "1". In returning from the break-interrupt state, the value transits from "1" to "0". The flag register **456** constitutes the return operation specifying section of the present invention.

The present state register (PSR) **457** holds the current processor state.

FIG. **7** is a representation showing an example of state transition of the processor in the first embodiment.

Referring to FIG. **7**, reference numeral **201** denotes a user state (to be referred to as a normal state hereinafter) without a normal interrupt or a break-interrupt; **202** denotes a supervisor state (to be referred to as a normal interrupt state hereinafter) without any break-interrupt; and **203** and **204** denote supervisor states (to be referred to as a break-interrupt state hereinafter) having a break-interrupt. When the processor is processing a normal application, the processor state is the normal state **201**.

When a normal interrupt **211** occurs in the normal state **201**, the processor transits to the normal interrupt state **202**. When a normal interrupt return instruction **212** is executed in this normal interrupt state **202**, the processor returns to the normal state **201**. When a break-interrupt **213** or **215** occurs in the normal state **201** or normal interrupt state **202**, the processor transits to the break-interrupt state **203** or **204**. When a break-interrupt return instruction **214** or **216** is executed in the break-interrupt state **203** or **204**, the processor returns to the normal state **201** or normal interrupt state **202** as the state before the break-interrupt.

FIG. **8** is a representation showing another example of state transition of the processor in the first embodiment.

Referring to FIG. **8**, reference numeral **250** denotes a normal state of the processor; **251**, **252**, and **253** denote normal interrupt states of the processor; and **254**, **255**, **256**, and **257** denote break-interrupt states of the processor. When the processor is processing a normal application, the processor state is the normal state **250**.

When a normal interrupt **261** occurs in the normal state **250**, the processor transits to the normal interrupt state **251**. When a normal interrupt return instruction **262** is executed in the normal interrupt state **251**, the processor returns to the normal state **250**. If another normal interrupt **265** occurs before execution of the normal interrupt return instruction **262** in the normal interrupt state **251**, the processor transits from the normal interrupt state **251** to the next normal interrupt state **252**. When a normal interrupt return instruction **266** is executed in this normal interrupt state **252**, the processor returns to the previous normal interrupt state **251**.

When a break-interrupt **263**, **267**, **269**, or **271** occurs in the normal state **250** or normal interrupt state **251**, **252**, and **253**, the processor transits to the break-interrupt state **254**, **255**, **256**, or **257**. When a break-interrupt return instruction **264**, **268**, **270**, or **272** is executed in the break-interrupt state **254**, **255**, **256**, or **257**, the processor returns to the previous normal state **250** or normal interrupt state **251**, **252**, or **253**.

The operation of the interrupt control apparatus shown in FIG. **6** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor is in the normal state **201**, the instruction fetch controller **421** shown in FIG. **6** reads out the instruction word **472** from the memory **410** on the basis of the instruction address **471** indicated by the program counter **422** and writes/holds the read-out instruction word in the instruction register **423**. The instruction fetch section **420** also supplies the instruction word **473** held in the instruction register **423** to the instruction word decoder **431**.

The instruction word decoder **431** decodes the received instruction word **473** and supplies an instruction to the instruction execution controller **432** or the breakpoint controller **433** in accordance with the decoding result. The instruction execution controller **432** or the breakpoint controller **433** executes processing in accordance with the received instruction. If the break-interrupt **213** or the normal interrupt **211** does not occur in this normal state **201**, the processor repeats the above operation.

However, when the instruction fetch controller **421** or the instruction execution controller **432** detects the normal interrupt **211**, the normal interrupt controller **441** is notified of the normal interrupt by the normal interrupt notification signal **479** from the instruction fetch controller **421** or the normal interrupt notification signal **486** from the instruction execution controller **432**. When receiving the normal interrupt notification from the instruction fetch controller **421** or the instruction execution controller **432**, the normal interrupt controller **441** controls the instruction fetch section **420** and the register section **450** to perform processing as follows.

First, the normal interrupt controller **441** reads out the presently indicated instruction address value **488** from the program counter **422** and writes the read-out instruction address value **488** in the normal return address register **452**. The normal interrupt controller **441** also reads out the processor state (normal state) before the normal interrupt from the present state register **457**, writes the read-out processor state in the normal previous state register **453**, and also writes the factor of the normal interrupt in the normal factor register **454**.

Next, the normal interrupt controller **441** writes, in the present state register **457**, the processor state that has transited in accordance with the normal interrupt. The normal interrupt controller **441** also supplies the start address **476** of the interrupt handler corresponding to the normal interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. Note that the flag register **456** keeps the initial value "0". Through the above-described process, the processor transits from the normal state **201** to the normal interrupt state **202**.

	Document ID	U	Title	Current OR
56	US 6549930 B1	☒	Method for scheduling threads in a multithreaded processor	718/104
57	US 6539467 B1	☒	Microprocessor with non-aligned memory access	711/219
58	US 6526421 B1	☒	Method of scheduling garbage collection	707/206
59	US 6490716 B1	☒	Automated design of processor instruction units	716/18
60	US 6453405 B1	☒	Microprocessor with non-aligned circular addressing	711/201
61	US 6449712 B1	☒	Emulating execution of smaller fixed-length branch/delay slot instructions with a sequence of larger fixed-length instructions	712/227
62	US 6430677 B2	☒	Methods and apparatus for dynamic instruction controlled reconfigurable register file with extended precision	712/210
63	US 6425100 B1	☒	Snoopy test access port architecture for electronic circuits including embedded core with built-in test access port	714/724
64	US 6415376 B1	☒	Apparatus and method for issue grouping of instructions in a VLIW processor	712/24
65	US 6408428 B1	☒	Automated design of processor systems using feedback from internal measurements of candidate systems	716/17
66	US 6397324 B1	☒	Accessing tables in memory banks using load and store address generators sharing store read port of compute register file separated from address register file	712/225
67	US 6393549 B1	☒	Instruction alignment unit for routing variable byte-length instructions	712/204
68	US 6385757 B1	☒	Auto design of VLIW processors	716/1
69	US 6381717 B1	☒	Snoopy test access port architecture for electronic circuits including embedded core having test access port with instruction driven wake-up	714/724
70	US 6378090 B1	☒	Hierarchical test access port architecture for electronic circuits including embedded core having built-in test access port	714/724
71	US 6360313 B1	☒	Instruction cache associative crossbar switch	712/215
72	US 6360312 B1	☒	Processor for making more efficient use of idling components and program conversion apparatus for the same	712/215
73	US 6351805 B1	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/219
74	US 6351802 B1	☒	Method and apparatus for constructing a pre-scheduled instruction cache	712/215
75	US 6347344 B1	☒	Integrated multimedia system with local processor, data transfer switch, processing modules, fixed functional unit, data streamer, interface unit and multiplexer, all integrated on multimedia processor	710/20
76	US 6343356 B1	☒	Methods and apparatus for dynamic instruction controlled reconfiguration register file with extended precision	712/210
77	US 6341343 B1	☒	Parallel processing instructions routed through plural differing capacity units of operand address generators coupled to multi-ported memory and ALUs	712/21

The processor which has transited to the normal interrupt state **202** reads out the instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **476** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction word decoder **431**.

The instruction word decoder **431** decodes the received instruction word **473** and supplies the instruction to one of the instruction execution controller **432**, the breakpoint controller **433**, and the interrupt return controller **434** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. Unless the break-interrupt **215** occurs in the normal interrupt state **202**, the instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

However, when the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433** detects the break-interrupt **215** during this operation, the break-interrupt controller **442** is notified of the break-interrupt by the break-interrupt notification signal **478** from the instruction fetch controller **421**, the break-interrupt notification signal **485** from the instruction execution controller **432**, or the break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from the program counter **422**, writes the read-out instruction address value **489** in the break return address register **455**, and also writes "1" in the flag register **456**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies the start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. Through the above-described process, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out the instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in the instruction word register **423**, and then supplies the read-out instruction word **473** to the instruction word decoder **431**.

The instruction word decoder **431** decodes the received instruction word **473** and supplies the instruction to the instruction execution controller **432** or the interrupt return controller **434** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. The instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes the break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from the memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434**.

When receiving the interrupt return instruction, the interrupt return controller **434** refers to the flag register **456** in the register section **450** and determines whether the flag register **456** has the value "1" representing the break-interrupt state. If the value of the flag register **456** is "1", the interrupt return controller **434** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

First, the interrupt return controller **434** writes "0" in the flag register **456**. The interrupt return controller **434** also reads out, from the break return address register **455**, the original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of the original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining parts of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes the normal interrupt return instruction **212**. At this time, the interrupt return instruction read out from the memory **410** and supplied to the instruction word decoder **431** by the instruction fetch controller **421** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434**.

When receiving the interrupt return instruction, the interrupt return controller **434** refers to the flag register **456** in the register section **450** and determines whether the flag register **456** has the value "1". In this case, the flag register **456** has the value "0" representing the non-break-interrupt state. Hence, the interrupt return controller **434** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

The interrupt return controller **434** reads out the value of the normal previous state register **453** and writes the value in the present state register **457**. The interrupt return controller **434** also reads out, from the normal return address register **452**, the original instruction address **475** to which the processor will return from the normal interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of the original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** for the normal operation from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies

	Document ID	U	Title	Current OR
78	US 6338137 B1	☒	Data processor having memory access unit with predetermined number of instruction cycles between activation and initial data transfer	712/225
79	US 6317821 B1	☒	Virtual single-cycle execution in pipelined processors	712/200
80	US 6317820 B1	☒	Dual-mode VLIW architecture providing a software-controlled varying mix of instruction-level and task-level parallelism	712/32
81	US 6304954 B1	☒	Executing multiple instructions in multi-pipelined processor by dynamically switching memory ports of fewer number than the pipeline	712/215
82	US 6282585 B1	☒	Cooperative interconnection for reducing port pressure in clustered microprocessors	710/5
83	US 6282505 B1	☒	Multi-port memory and a data processor accessing the same	703/25
84	US 6269440 B1	☒	Accelerating vector processing using plural sequencers to process multiple loop iterations simultaneously	712/241
85	US 6269437 B1	☒	Duplicator interconnection methods and apparatus for reducing port pressure in a clustered processor	712/28
86	US 6256687 B1	☒	Managing data flow between a serial bus device and a parallel port	710/71
87	US 6247115 B1	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/219
88	US 6240508 B1	☒	Decode and execution synchronized pipeline processing using decode generated memory read queue with stop entry to allow execution generated memory read	712/219
89	US 6237073 B1	☒	Method for providing virtual memory to physical memory page mapping in a computer operating system that randomly samples state information	711/202
90	US 6230251 B1	☒	File replication methods and apparatus for reducing port pressure in a clustered processor	712/11
91	US 6191713 B1	☒	Conversion between serial bus cycles and parallel port commands using a state machine	341/100
92	US 6189088 B1	☒	Forwarding stored dara fetched for out-of-order load/read operation to over-taken operation read-accessing same memory location	712/216
93	US 6185629 B1	☒	Data transfer controller employing differing memory interface protocols dependent upon external input at predetermined time	710/10
94	US 6175910 B1	☒	Speculative instructions exection in VLIW processors	712/217
95	US 6173356 B1	☒	Multi-port DRAM with integrated SRAM and systems and methods using the same	711/5
96	US 6163839 A	☒	Non-stalling circular counterflow pipeline processor with reorder buffer	712/219
97	US 6163837 A	☒	Writing of instruction results produced by instruction execution circuits to result destinations	712/216
98	US 6145025 A	☒	Method for transferring DMA data in a multimedia intergrated circuit including preloading DMA instructions in a frame buffer	710/22
99	US 6141746 A	☒	Information processor	712/214
100	US 6122722 A	☒	VLIW processor with less instruction issue slots than functional units	712/24

the instruction word to the instruction execution section **430**. Thus, the processor transits from the normal interrupt state **202** to the normal state **201**. The instruction execution section **430** executes the remaining part of processing of the application corresponding to the normal operation.

As described above in detail, according to this embodiment, when a normal interrupt occurs, the processor operation information before the normal interrupt is saved by writing, in the normal return address register **452**, the original instruction address to which the processor will return from the normal interrupt state. When a break-interrupt occurs, the processor operation information before the break-interrupt is saved by writing, in the break return address register **455**, the original instruction address to which the processor will return from the break-interrupt state. In addition, whether a break-interrupt has occurred is set in the flag register **456**. In returning from the interrupt, which one of addresses of the return address registers **452** and **455** is to be used is determined by referring to the value of the flag register **456**.

According to this construction, even within the interrupt inhibition periods immediately after the interrupt operation by a normal interrupt and immediately before interrupt return, when a write in the normal return address register **452**, the normal previous state register **453**, and the normal factor register **454** is inhibited, the break return address can be written in the break return address register **455** different from the normal return address register **452**. Hence, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt. Additionally, the operation information before the normal interrupt or break-interrupt can be accurately restored by referring to the value of the flag register **456** in executing an interrupt return instruction.

Second Embodiment

The second embodiment of the present invention will be described next.

FIG. **9** is a block diagram showing the construction of an interrupt control apparatus according to the second embodiment. In FIG. **9**, the same reference numerals as in FIG. **6** denote the same blocks as in FIG. **6**, respectively, and a detailed description thereof will be omitted.

In the second embodiment shown in FIG. **9**, in addition to registers **451** to **457** shown in FIG. **6**, a break previous state register (BEPSR) **458** for holding the processor state before a break-interrupt is provided.

In this embodiment, the normal return address register **452**, the normal previous state register **453**, and the normal factor register **454** constitute the first information holding section of the present invention, and the break return address register **455** and the break previous state register **458** constitute the second information holding section of the present invention.

The flag register **456** constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **9** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and a register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422**, writes the read-out instruction address value **489** in the break return address register **455**, and also writes "1" in the flag register **456**. The break-interrupt controller **442** also reads out the processor state (normal interrupt state) before the break-interrupt from the present state register **457** and writes the read-out processor state in the break previous state register **458**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**.

The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or interrupt return controller **434** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434**.

When receiving the interrupt return instruction, the interrupt return controller **434** refers to the flag register **456** in the register section **450** and determines whether the flag register **456** has the value "1" representing the break-interrupt state if the value of the flag register **456** is "1", the interrupt return controller **434** controls the instruction fetch section **420** and the register section **450** to perform processing as follows.

First, the interrupt return controller **434** writes "0" in the flag register **456** and simultaneously reads out the value of

	Document ID	U	Title	Current OR
101	US 6119195 A	☒	Virtualizing serial bus information point by address mapping via a parallel port	710/310
102	US 6092180 A	☐	Method for measuring latencies by randomly selected sampling of the instructions while the instruction are executed	712/200
103	US 6076146 A	☒	Cache holding register for delayed update of a cache line into an instruction cache	711/125
104	US 6065106 A	☒	Resuming normal execution by restoring without refetching instructions in multi-word instruction register interrupted by debug instructions loading and processing	712/24
105	US 6065105 A	☒	Dependency matrix	712/23
106	US 6055649 A	☒	Processor test port with scan chains and data streaming	714/30
107	US 6047351 A	☒	Jitter free instruction execution	710/266
108	US 6044451 A	☒	VLIW processor with write control unit for allowing less write buses than functional units	712/24
109	US 6044222 A	☒	System, method, and program product for loop instruction scheduling hardware lookahead	717/156
110	US 6038396 A	☒	Compiling apparatus and method for a VLIW system computer and a recording medium for storing compile execution programs	717/161
111	US 6016540 A	☒	Method and apparatus for scheduling instructions in waves	712/214
112	US 6006324 A	☒	High performance superscalar alignment unit	712/204
113	US 6002880 A	☒	VLIW processor with less instruction issue slots than functional units	712/24
114	US 5983321 A	☒	Cache holding register for receiving instruction packets and for providing the instruction packets to a predecode unit and instruction cache	711/125
115	US 5978907 A	☒	Delayed update register for an array	712/239
116	US 5974537 A	☒	Guard bits in a VLIW instruction control routing of operations to functional units allowing two issue slots to specify the same functional unit	712/215
117	US 5968167 A	☒	Multi-threaded data processing management system	712/225
118	US 5966530 A	☒	Structure and method for instruction boundary machine state restoration	712/244
119	US 5964867 A	☒	Method for inserting memory prefetch operations based on measured latencies in a program optimizer	712/219
120	US 5958048 A	☒	Architectural support for software pipelining of nested loops	712/241
121	US 5950007 A	☒	Method for compiling loops containing prefetch instructions that replaces one or more actual prefetches with one virtual prefetch prior to loop scheduling and unrolling	717/161
122	US 5943064 A	☒	Apparatus for processing multiple types of graphics data for display	345/546
123	US 5941983 A	☒	Out-of-order execution using encoded dependencies between instructions in queues to determine stall values that control issurance of instructions from the queues	712/214

the break previous state register **458** and writes the value in the present state register **457**. The interrupt return controller **434** also reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212** and returns from the normal interrupt state **202** to the normal state **201**. The operation at this time is the same as that of the interrupt control apparatus shown in FIG. **6**.

As described above, in the second embodiment, in addition to the operation of the above-described first embodiment, when a break-interrupt occurs, the processor state before the break-interrupt is written in the break previous state register **458**.

According to this construction, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt, and additionally, the processor state before the interrupt can be held even for the break-interrupt. In returning from the break-interrupt, the previous processor state can be easily restored only by referring to the value of one register.

Third Embodiment

The third embodiment of the present invention will be described next.

FIG. **10** is a block diagram showing the construction of an interrupt control apparatus according to the third embodiment. In FIG. **10**, the same reference numerals as in FIG. **6** denote the same blocks as in FIG. **6**, respectively, and a detailed description thereof will be omitted.

In the third embodiment shown in FIG. **10**, in addition to the registers **451** to **457** shown in FIG. **6**, a break factor register (BECR) **459** for holding the factor (instruction breakpoint, data breakpoint, software breakpoint, or step execution) of a break-interrupt is provided.

In this embodiment, the normal return address register **452**, the normal previous state register **453**, and the normal factor register **454** constitute the first information holding section of the present invention, and the break return address register **455** and the break factor register **459** constitute the second information holding section of the present invention.

The flag register **456** constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **10** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and the register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422**, writes the read-out instruction address value **489** in the break return address register **455**, and also writes "1" in the flag register **456**. The break-interrupt controller **442** also writes the factor of the break-interrupt in the break factor register **459**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**.

The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or interrupt return controller **434** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434**.

When receiving the interrupt return instruction, the interrupt return controller **434** refers to the flag register **456** in the register section **450** and determines whether the flag register **456** has the value "1" representing the break-interrupt state. If the value of the flag register **456** is "1", the interrupt return controller **434** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

	Document ID	U	Title	Current OR
124	US 5931939 A	☒	Read crossbar elimination in a VLIW processor	712/24
125	US 5925097 A	☒	Directly programmable distribution element	709/200
126	US 5923863 A	☒	Software mechanism for accurately handling exceptions generated by instructions scheduled speculatively due to branch elimination	712/216
127	US 5913049 A	☒	Multi-stream complex instruction set microprocessor	712/215
128	US 5887174 A	☒	System, method, and program product for instruction scheduling in the presence of hardware lookahead accomplished by the rescheduling of idle slots	717/161
129	US 5840060 A	☒	Processor which performs dynamic instruction scheduling at time of execution within a single clock cycle	712/215
130	US 5878267 A	☒	Compressed instruction format for use in a VLIW processor and processor for processing such instructions	712/24
131	US 5878255 A	☒	Update unit for providing a delayed update to a branch prediction array	712/240
132	US 5862399 A	☒	Write control unit	712/24
133	US 5852741 A	☒	VLIW processor which processes compressed instruction format	712/24
134	US 5835776 A	☒	Method and apparatus for instruction scheduling in an optimizing compiler for minimizing overhead instructions	717/161
135	US 5832249 A	☒	High performance superscalar alignment unit	712/204
136	US 5826054 A	☒	Compressed Instruction format for use in a VLIW processor	712/213
137	US 5822579 A	☒	Microprocessor with dynamically controllable microcontroller condition selection	712/245
138	US 5822559 A	☒	Apparatus and method for aligning variable byte-length instructions to a plurality of issue positions	712/214
139	US 5822558 A	☒	Method and apparatus for predecoding variable byte-length instructions within a superscalar microprocessor	712/213
140	US 5819088 A	☒	Method and apparatus for scheduling instructions for execution on a multi-issue architecture computer	717/149
141	US 5819059 A	☒	Predecode unit adapted for variable byte-length instruction set processors and method of operating the same	712/213
142	US 5819057 A	☒	Superscalar microprocessor including an instruction alignment unit with limited dispatch to decode units	712/204
143	US 5815696 A	☒	Pipeline processor including interrupt control system for accurately perform interrupt processing even applied to VLIW and delay branch instruction in delay slot	712/233
144	US 5802575 A	☒	Hit bit for indicating whether load buffer entries will hit a cache when they reach buffer head	711/144
145	US 5794029 A	☒	Architectural support for execution control of prologue and eplogue periods of loops in a VLIW processor	712/241
146	US 5794003 A	☒	Instruction cache associative crossbar switch system	712/215

First, the interrupt return controller **434** writes "0" in the flag register **456**. The interrupt return controller **434** also reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212** and returns from the normal interrupt state **202** to the normal state **201**. The operation at this time is the same as that of the interrupt control apparatus shown in FIG. **6**.

As described above, in the third embodiment, in addition to the operation of the above-described first embodiment, when a break-interrupt occurs, the factor of the break-interrupt is written in the break factor register **459**.

According to this construction, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt, and additionally, the interrupt factor can be held even for the break-interrupt. For this reason, in a break-interrupt handler, not only predetermined specific processing but also appropriate interrupt processing corresponding to the interrupt factor can be performed.

Fourth Embodiment

The fourth embodiment of the present invention will be described next.

FIG. **11** is a block diagram showing the construction of an interrupt control apparatus according to the fourth embodiment. In FIG. **11**, the same reference numerals as in FIGS. **6**, **9**, and **10** denote the same blocks as in FIGS. **6**, **9**, and **10**, respectively, and a detailed description thereof will be omitted.

In the fourth embodiment shown in FIG. **11**, in addition to the registers **451** to **457** shown in FIG. **6**, a break previous state register **458** and a break factor register **459** are provided.

In this embodiment, the normal return address register **452**, the normal previous state register **453**, and the normal factor register **454** constitute the first information holding section of the present invention, and the break return address register **455**, the break previous state register **458**, and the break factor register **459** constitute the second information holding section of the present invention.

The flag register **456** constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **11** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and a register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422**, writes the read-out instruction address value **489** in the break return address register **455**, and also writes "1" in the flag register **456**. The break-interrupt controller **442** also reads out the processor state (normal interrupt state) before the break-interrupt from the present state register **457** and writes the read-out processor state in the break previous state register **458**, and simultaneously writes the factor of the break-interrupt in the break factor register **459**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**. The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or the interrupt return controller **434** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434**.

When receiving the interrupt return instruction, the interrupt return controller **434** refers to the flag register **456** in the

	Document ID	U	Title	Current OR
147	US 5787302 A	☒	Software for producing instructions in a compressed format for a VLIW processor	712/24
148	US 5761475 A	☒	Computer processor having a register file with reduced read and/or write port bandwidth	712/218
149	US 5751985 A	☒	Processor structure and method for tracking instruction status to maintain precise state	712/218
150	US 5745729 A	☒	Methods and apparatuses for servicing load instructions	711/131
151	US 5712999 A	☒	Address generator employing selective merge of two independent addresses	711/211
152	US 5699536 A	☒	Computer processing system employing dynamic instruction formatting	712/216
153	US 5692139 A	☒	VLIW processing device including improved memory for avoiding collisions without an excessive number of ports	710/316
154	US 5673426 A	☒	Processor structure and method for tracking floating-point exceptions	712/244
155	US 5659721 A	☒	Processor structure and method for checkpointing instructions to maintain precise state	712/228
156	US 5655133 A	☒	Massively multiplexed superscalar Harvard architecture computer	712/23
157	US 5655115 A	☒	Processor structure and method for watchpoint of plural simultaneous unresolved branch evaluation	712/239
158	US 5655096 A	☒	Method and apparatus for dynamic scheduling of instructions to ensure sequentially coherent data in a processor employing out-of-order execution	712/200
159	US 5651124 A	☒	Processor structure and method for aggressively scheduling long latency instructions including load/store instructions while maintaining precise state	712/215
160	US 5649136 A	☒	Processor structure and method for maintaining and restoring precise state at any instruction boundary	712/244
161	US 5644780 A	☒	Multiple port high speed register file with interleaved write ports for use with very long instruction word (vlin) and n-way superscaler processors	712/23
162	US 5644742 A	☒	Processor structure and method for a time-out checkpoint	712/244
163	US 5640588 A	☒	CPU architecture performing dynamic instruction scheduling at time of execution within single clock cycle	712/23
164	US 5634023 A	☒	Software mechanism for accurately handling exceptions generated by speculatively scheduled instructions	712/244
165	US 5634004 A	☒	Directly programmable distribution element	710/317
166	US 5627982 A	☒	Apparatus for simultaneously scheduling instructions from plural instruction stream into plural instruction executions units	712/206
167	US 5627981 A	☒	Software mechanism for accurately handling exceptions generated by instructions scheduled speculatively due to branch elimination	712/235
168	US 5619665 A	☒	Method and apparatus for the transparent emulation of an existing instruction-set architecture by an arbitrary underlying instruction-set architecture	712/208
169	US 5598546 A	☒	Dual-architecture super-scalar pipeline	712/209

register section **450** and determines whether the flag register **456** has the value "1" representing the break-interrupt state. If the value of the flag register **456** is "1", the interrupt return controller **434** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

First, the interrupt return controller **434** writes "0" in the flag register **456** and simultaneously reads out the value of the break previous state register **458** and writes the value in the present state register **457**. The interrupt return controller **434** also reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212** and returns from the normal interrupt state **202** to the normal state **201**. The operation at this time is the same as that of the interrupt control apparatus shown in FIG. **6**.

As described above, in the fourth embodiment, in addition to the operation of the above-described first embodiment, when a break-interrupt occurs, the processor state before the break-interrupt is written in the break previous state register **458**, and the factor of the break-interrupt is written in the break factor register **459**.

According to this construction, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt. In addition, even in a break-interrupt handler, the previous processor state can be easily restored only in returning from the break-interrupt, and appropriate interrupt processing corresponding to the break-interrupt factor can be performed.

Fifth Embodiment

The fifth embodiment of the present invention will be described next.

FIG. **12** is a block diagram showing the construction of an interrupt control apparatus according to the fifth embodiment. In FIG. **12**, the same reference numerals as in FIG. **6** denote the same blocks as in FIG. **6**, respectively, and a detailed description thereof will be omitted.

Referring to FIG. **12**, an interrupt return controller **434'** executes a return operation from an interrupt state, like the interrupt return controller **434** shown in FIG. **6**. The interrupt return controller **434'** of this embodiment specifies whether the operation is a return operation from a normal interrupt state or a return operation from a break-interrupt state on the basis of information in the interrupt return instruction and restores operation information before the interrupt.

Additionally, in this embodiment, the flag register **456** used in the first to fourth embodiments is omitted, and instead, an interrupt return instruction (to be described later) contains information on whether a break-interrupt state is set.

In this embodiment, a normal return address register **452**, a normal previous state register **453**, and a normal factor register **454** constitute the first information holding section of the present invention, and a break return address register **455** constitutes the second information holding section of the present invention.

FIG. **13** is a representation showing the instruction form of an interrupt return instruction according to this embodiment.

Referring to FIG. **13**, reference numeral **101** denotes a field representing an instruction code; and **102** denotes a field representing an operand. In this embodiment, the instruction code **101** means an interrupt return instruction, and the value of the operand **102** means whether a break-interrupt state is set. The operand **102** having the value "0" means return from a normal interrupt state, and the value "1" means return from a break-interrupt state.

The interrupt return instruction shown in FIG. **13** is prepared at the end of each of an interrupt handler for a normal interrupt and an interrupt handler for a break-interrupt. The value of the operand **102** of the interrupt return instruction for a normal interrupt is set to "0", and the value of the operand **102** of the interrupt return instruction for a break-interrupt is set to "1". The operand **102** of the interrupt return instruction constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **12** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422** and writes the read-out instruction address value **489** in the break return address register **455**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt

	Document ID	U	Title	Current OR
170	US 5574939 A	☒	Multiprocessor coupling system with integrated compile and run time scheduling for parallelism	712/24
171	US 5560028 A	☒	Software scheduled superscalar computer architecture	712/23
172	US 5555384 A	☒	Rescheduling conflicting issued instructions by delaying one conflicting instruction into the same pipeline stage as a third non-conflicting instruction	712/216
173	US 5542058 A	☒	Pipelined computer with operand context queue to simplify context-dependent execution flow	713/502
174	US 5537561 A	☒	Processor	712/23
175	US 5488729 A	☒	Central processing unit architecture with symmetric instruction scheduling to achieve multiple instruction launch and execution	712/209
176	US 5488709 A	☒	Cache including decoupling register circuits	711/118
177	US 5481736 A	☒	Computer processing element having first and second functional units accessing shared memory output port on prioritized basis	712/23
178	US 5471593 A	☒	Computer processor with an efficient means of executing many instructions simultaneously	712/235
179	US 5471591 A	☒	Combined write-operand queue and read-after-write dependency scoreboard	712/217
180	US 5450556 A	☒	VLIW processor which uses path information generated by a branch control unit to inhibit operations which are not on a correct path	712/235
181	US 5434972 A	☒	Network for determining route through nodes by directing searching path signal arriving at one port of node to another port receiving free path signal	709/238
182	US 5430851 A	☒	Apparatus for simultaneously scheduling instruction from plural instruction streams into plural instruction execution units	712/212
183	US 5428811 A	☒	Interface between a register file which arbitrates between a number of single cycle and multiple cycle functional units	712/23
184	US 5404555 A	☒	Macro instruction set computer architecture	712/36
185	US 5404469 A	☒	Multi-threaded microprocessor architecture utilizing static interleaving	712/215
186	US 5355335 A	☒	Semiconductor memory device having a plurality of writing and reading ports for decreasing hardware amount	365/189.04
187	US 5329630 A	☒	System and method using double-buffer preview mode	711/173
188	US 5135151 A	☒	Multiport memory bypass under software control	711/149
189	US 5299321 A	☒	Parallel processing device to operate with parallel execute instructions	712/212
190	US H001291 H	☒	Microprocessor in which multiple instructions are executed in one clock cycle by providing separate machine bus access to a register file for different types of instructions	712/23
191	US 5197137 A	☒	Computer architecture for the concurrent execution of sequential programs	718/107
192	US 5168276 A	☒	Automatic A/D converter operation using a programmable control table	341/141

handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**.

The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or the interrupt return controller **434'** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434'**.

When receiving the interrupt return instruction, the interrupt return controller **434'** refers to the operand **102** of the received interrupt return instruction and determines whether the value is "1". If the operand **102** has the value "1" representing return from the break-interrupt state, the interrupt return controller **434'** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

First, the interrupt return controller **434'** reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212**. At this time, the interrupt return instruction read out from the memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434'**.

When receiving the interrupt return instruction, the interrupt return controller **434'** refers to the operand **102** of the received interrupt return instruction and determines whether the value is "1". In this case, the operand **102** has the value "0" representing return from the normal interrupt state. The interrupt return controller **434'** controls the instruction fetch section **420** and the register section **450** to start the normal interrupt return operation. In this normal interrupt return operation, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed, and the normal interrupt state **202** transits to the normal state **201**.

As described above, according to the fifth embodiment, when a normal interrupt occurs, the processor operation information before the normal interrupt is saved by writing, in the normal return address register **452**, the original instruction address to which the processor will return from the normal interrupt state. When a break-interrupt occurs, the processor operation information before the break-interrupt is saved by writing, in the break return address register **455**, the original instruction address to which the processor will return from the break-interrupt state. In addition, an interrupt return instruction whose operand **102** has the value "0" is prepared at the end of an interrupt handler corresponding to a normal interrupt, and an interrupt return instruction whose operand **102** has the value "1" is prepared at the end of an interrupt handler corresponding to a break-interrupt. In returning from the interrupt, the address to be used, i.e., the address in the return address register **452** or **455** is determined by referring to the value of the operand **102**.

According to this construction, even within the interrupt inhibition periods immediately after the interrupt operation by a normal interrupt and immediately before interrupt return, the break return address can be written in the break return address register **455** different from the normal return address register **452**. Hence, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt. Additionally, the operation information before the normal interrupt or break-interrupt can be accurately restored by referring to the value of the operand **102** of an interrupt return instruction in executing the interrupt return instruction. Furthermore, since not a register but the operand **102** in the interrupt return instruction has an identifier representing return from the normal interrupt state or return from the break-interrupt state, the processor operation information before the interrupt can be restored with a minimum hardware resource.

Sixth Embodiment

The sixth embodiment of the present invention will be described next.

FIG. **14** is a block diagram showing the construction of an interrupt control apparatus according to the sixth embodiment. In FIG. **14**, the same reference numerals as in FIGS. **6**, **9**, and **12** denote the same blocks as in FIGS. **6**, **9**, and **12**, respectively, and a detailed description thereof will be omitted.

In this embodiment, an interrupt return instruction shown in FIG. **13** contains information on whether a break-interrupt state is set, like in the fifth embodiment shown in FIG. **12**.

In this embodiment, a normal return address register **452**, a normal previous state register **453**, and a normal factor register **454** constitute the first information holding section of the present invention, and a break return address register **455** and a break previous state register **458** constitute the second information holding section of the present invention.

The interrupt return instruction is prepared at the end of either of an interrupt handler for a normal interrupt and an interrupt handler for a break-interrupt. The value of an operand **102** of the interrupt return instruction for a normal interrupt is set to "0", and the value of the operand **102** of

	Document ID	U	Title	Current OR
193	US 4845659 A	☒	Accelerated validity response permitting early issue of instructions dependent upon outcome of floating point operations	712/222
194	US 4626985 A	☒	Single-chip microcomputer with internal time-multiplexed address/data/interrupt bus	712/40
195	US 4325120 A	☒	Data processing system	711/202
196	US 4017840 A	☒	Method and apparatus for protecting memory storage location accesses	711/164
197	US 3916384 A	☒	Communication switching system computer memory control arrangement	711/149
198	US 3916112 A	☒	Stored program control with memory work area assignment in a communication switching system	379/244
199	US 3845425 A	☒	METHOD AND APPARATUS FOR PROVIDING CONDITIONAL AND UNCONDITIONAL ACCESS TO PROTECTED MEMORY STORAGE LOCATIONS	711/152

the interrupt return instruction for a break-interrupt is set to "1". The operand **102** of the interrupt return instruction constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **14** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422** and writes the read-out instruction address value **489** in the break return address register **455**. The break-interrupt controller **442** also reads out the processor state (normal interrupt state) before the break-interrupt from the present state register **457** and writes the read-out processor state in the break previous state register **458**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**. The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or interrupt return controller **434'** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434'**.

When receiving the interrupt return instruction, the interrupt return controller **434'** refers to the operand **102** of the received interrupt return instruction and determines whether the value is "1". If the operand **102** has the value "1" representing return from the break-interrupt state, the interrupt return controller **434'** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

First, the interrupt return controller **434'** reads out the value of the break previous state register **458** and writes the value in the present state register **457**. The interrupt return controller **434'** also reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212** and returns from the normal interrupt state **202** to the normal state **201**. The operation at this time is the same as that of the interrupt control apparatus shown in FIG. **12**.

As described above, in the sixth embodiment, in addition to the operation of the above-described fifth embodiment, when a break-interrupt occurs, the processor state before the break-interrupt is written in the break previous state register **458**.

According to this construction, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt, and additionally, the processor state before the interrupt can be held even for the break-interrupt. In returning from the break-interrupt, the previous processor state can be easily restored.

Seventh Embodiment

The seventh embodiment of the present invention will be described next.

FIG. **15** is a block diagram showing the construction of an interrupt control apparatus according to the seventh embodiment. In FIG. **15**, the same reference numerals as in FIGS. **6**, **10**, and **12** denote the same blocks as in FIGS. **6**, **10**, and **12**, respectively, and a detailed description thereof will be omitted.

In this embodiment, an interrupt return instruction shown in FIG. **13** contains information on whether a break-interrupt state is set, like in the fifth embodiment shown in FIG. **12**.

In this embodiment, a normal return address register **452**, a normal previous state register **453**, and a normal factor

	Document ID	U	Title	Current OR
1	US 20030232648 A1	☐	Videophone and videoconferencing apparatus and method for a video game console	463/40
2	US 20030229780 A1	☒	Multiconfiguable device masking shunt and method of use	713/153
3	US 20030212897 A1	☒	Method and system for maintaining secure semiconductor device areas	713/200
4	US 20030212830 A1	☒	Communications system using rings architecture	709/251
5	US 20030206409 A1	☒	Miniature flashlight having replaceable battery pack and multiple operating modes	362/183
6	US 20030204819 A1	☒	Method of generating development environment for developing system LSI and medium which stores program therefor	716/1
7	US 20030204636 A1	☒	Communications system using rings architecture	709/251
8	US 20030200422 A1	☒	Parallel processor	712/215
9	US 20030200351 A1	☒	Method frame storage using multiple memory circuits	719/315
10	US 20030200343 A1	☒	Communications system using rings architecture	709/251
11	US 20030200342 A1	☒	Communications system using rings architecture	709/251
12	US 20030200339 A1	☒	Communications system using rings architecture	709/250
13	US 20030196076 A1	☒	Communications system using rings architecture	712/234
14	US 20030195991 A1	☒	Communications system using rings architecture	709/251
15	US 20030195990 A1	☒	Communications system using rings architecture	709/251
16	US 20030195989 A1	☒	Communications system using rings architecture	709/251
17	US 20030191863 A1	☒	Communications system using rings architecture	709/251

register **454** constitute the first information holding section of the present invention, and a break return address register **455** and a break factor register **459** constitute the second information holding section of the present invention.

The interrupt return instruction is prepared at the end of either of an interrupt handler for a normal interrupt and an interrupt handler for a break-interrupt. The value of an operand **102** of the interrupt return instruction for a normal interrupt is set to "0", and the value of the operand **102** of the interrupt return instruction for a break-interrupt is set to "1". The operand **102** of the interrupt return instruction constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **15** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422** and writes the read-out instruction address value **489** in the break return address register **455**. The break-interrupt controller **442** also writes the factor of the break-interrupt in the break factor register **459**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**. The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or the interrupt return controller **434'** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instruction. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434'**.

When receiving the interrupt return instruction, the interrupt return controller **434'** refers to the operand **102** of the received interrupt return instruction and determines whether the value is "1". If the operand **102** has the value "1" representing return from the break-interrupt state, the interrupt return controller **434'** controls the instruction fetch section **420** and register section **450** to perform processing as follows.

First, the interrupt return controller **434'** reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**. On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212** and returns from the normal interrupt state **202** to the normal state **201**. The operation at this time is the same as that of the interrupt control apparatus shown in FIG. **12**.

As described above, in the seventh embodiment, in addition to the operation of the above-described fifth embodiment, when a break-interrupt occurs, the factor of the break-interrupt is written in the break factor register **459**.

According to this construction, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt, and additionally, the interrupt factor can be held even for the break-interrupt. For this reason, in a break-interrupt handler, not only predetermined specific processing but also appropriate interrupt processing corresponding to the interrupt factor can be performed.

Eighth Embodiment

The eighth embodiment of the present invention will be described next.

FIG. **16** is a block diagram showing the construction of an interrupt control apparatus according to the eighth embodiment. In FIG. **16**, the same reference numerals as in FIGS. **6**, **9**, **10**, and **12** denote the same blocks as in FIGS. **6**, **9**, **10**, and **12**, respectively, and a detailed description thereof will be omitted.

In this embodiment, an interrupt return instruction shown in FIG. **13** contains information on whether a break-interrupt state is set, like in the fifth embodiment shown in FIG. **12**.

	Document ID	U	Title	Current OR
18	US 20030191862 A1	☒	Communications system using rings architecture	709/251
19	US 20030191861 A1	☒	Communications system using rings architecture	709/251
20	US 20030189940 A1	☒	Communications system using rings architecture	370/406
21	US 20030174252 A1	☒	Programmable motion estimation module with vector array unit	348/699
22	US 20030172257 A1	☒	Communications system using rings architecture	712/234
23	US 20030172190 A1	☒	Communications system using rings architecture	709/251
24	US 20030172189 A1	☒	Communications system using rings architecture	709/251
25	US 20030172147 A1	☒	Application programming interfaces and methods enabling a host to interface with a network processor	709/223
26	US 20030167348 A1	☒	Communications system using rings architecture	709/251
27	US 20030115238 A1	☒	Method frame storage using multiple memory circuits	718/100
28	US 20030101371 A1	☒	Method, system, and program for error handling in a dual adaptor system where one adaptor is a master	714/9
29	US 20030093721 A1	☒	Selective automated power cycling of faulty disk in intelligent disk array enclosure for error recovery	714/42
30	US 20030077085 A1	☒	Image formation apparatus and control method thereof	399/16
31	US 20030067473 A1	☒	Method and apparatus for executing a predefined instruction set	345/561
32	US 20030061329 A1	☒	Method and system for logging data in a cellular data network	709/223
33	US 20030026298 A1	☒	Flexible multiplexer/demultiplexer and method for transport of optical line data to a wide/metro area link	370/537
34	US 20020194182 A1	☒	Computer system	707/10

In this embodiment, a normal return address register **452**, a normal previous state register **453**, and a normal factor register **454** constitute the first information holding section of the present invention, and a break return address register **455**, a break previous state register **458**, and a break factor register **459** constitute the second information holding section of the present invention.

The interrupt return instruction is prepared at the end of either of an interrupt handler for a normal interrupt and an interrupt handler for a break-interrupt. The value of an operand **102** of the interrupt return instruction for a normal interrupt is set to "0", and the value of the operand **102** of the interrupt return instruction for a break-interrupt is set to "1". The operand **102** of the interrupt return instruction constitutes the return operation specifying section of the present invention.

The operation of the interrupt control apparatus shown in FIG. **16** will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

When the processor in the normal state **201** transits to the normal interrupt state **202** due to a normal interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **6** is performed.

When the processor is in the normal interrupt state **202**, and an instruction fetch controller **421**, an instruction execution controller **432**, or an interrupt return controller **434** detects a break-interrupt **215**, a break-interrupt controller **442** is notified of the break-interrupt by a break-interrupt notification signal **478** from the instruction fetch controller **421**, a break-interrupt notification signal **485** from the instruction execution controller **432**, or a break-interrupt notification signal **487** from the breakpoint controller **433**.

When receiving the break-interrupt notification from the instruction fetch controller **421**, the instruction execution controller **432**, or the breakpoint controller **433**, the break-interrupt controller **442** controls an instruction fetch section **420** and a register section **450** to perform processing as follows.

First, the break-interrupt controller **442** reads out the currently indicated instruction address value **489** from a program counter **422** and writes the read-out instruction address value **489** in the break return address register **455**. The break-interrupt controller **442** also reads out the processor state (normal interrupt state) before the break-interrupt from the present state register **457** and writes the read-out processor state in the break previous state register **458**. The break-interrupt controller **442** also writes the factor of the break-interrupt in the break factor register **459**.

Next, the break-interrupt controller **442** writes, in the present state register **457**, the processor state that has transited in accordance with the break-interrupt. The break-interrupt controller **442** also supplies a start address **477** of the interrupt handler corresponding to the break-interrupt to the instruction fetch section **420** and sets the address value in the program counter **422**. By processing as described above, the processor transits from the normal interrupt state **202** to the break-interrupt state **204**.

The processor which has transited to the break-interrupt state **204** reads out an instruction word **472** of the interrupt handler to the instruction fetch controller **421** in accordance with the start address **477** of the interrupt handler, which is set in the program counter **422**, temporarily holds the read-out instruction word in an instruction word register **423**, and then supplies the read-out instruction word **473** to an instruction word decoder **431**.

The instruction word decoder **431** decodes a received instruction word **473** and supplies the instruction to the instruction execution controller **432** or the interrupt return controller **434'** in accordance with the decoding result. The controller which has received the instruction executes processing in accordance with the received instruction. An instruction execution section **430** repeats the operation of executing the instruction word **473** sequentially supplied toward the final address of the interrupt handler.

When processing of the interrupt handler corresponding to the break-interrupt is ended, the processor executes a break-interrupt return instruction **216**. At this time, the interrupt return instruction read out from a memory **410** and supplied to the instruction word decoder **431** by the instruction fetch section **420** is decoded by the instruction word decoder **431** and determined as an interrupt return instructions. In accordance with this determination, the instruction word decoder **431** supplies the interrupt return instruction to the interrupt return controller **434'**.

When receiving the interrupt return instruction, the interrupt return controller **434'** refers to the operand **102** of the received interrupt return instruction and determines whether the value is "1". If the operand **102** has the value "1" representing return from the break-interrupt state, the interrupt return controller **434'** controls the instruction fetch section **420** and the register section **450** to perform processing as follows.

First, the interrupt return controller **434'** reads out the value of the break previous state register **458** and writes the value in the present state register **457**. The interrupt return controller **434'** also reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in the program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, the instruction fetch controller **421** reads out the instruction word **472** of the interrupt handler corresponding to the normal interrupt from the memory **410**, temporarily holds the instruction word in the instruction word register **423**, and then supplies the instruction word to the instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes a normal interrupt return instruction **212** and returns from the normal interrupt state **202** to the normal state **201**. The operation at this time is the same as that of the interrupt control apparatus shown in FIG. **12**.

As described above, in the eighth embodiment, in addition to the operation of the above-described fifth embodiment, when a break-interrupt occurs, the processor state before the break-interrupt is written in the break previous state register **458**, and the factor of the break-interrupt is written in the break factor register **459**.

According to this construction, a break-interrupt can occur even within the interrupt inhibition period by a normal interrupt. In addition, even in a break-interrupt handler, the previous processor state can be easily restored only in returning from the break-interrupt, and appropriate interrupt processing corresponding to the break-interrupt factor can be performed.

	Document ID	U	Title	Current OR
35	US 20020191546 A1	☒	Digital subscriber line access and network testing multiplexer	370/252
36	US 20020176445 A1	☒	Radio communication arrangements	370/480
37	US 20020159806 A1	☒	Printing data and picture data transferring method	400/61
38	US 20020159092 A1	☒	Method and apparatus for embodying documents	358/1.15
39	US 20020152348 A1	☒	Method of configuring electronic devices	710/313
40	US 20020087846 A1	☒	Reconfigurable processing system and method	712/229
41	US 20020087687 A1	☒	System resource availability manager	709/225
42	US 20020046355 A1	☒	Electronic device and its power control method	713/320
43	US 20020032875 A1	☒	Information processing apparatus and method	713/300
44	US 20020032748 A1	☒	Communication apparatus detecting method	709/217
45	US 20020009301 A1	☒	Image formation apparatus and control method thereof	399/16
46	US 20020004916 A1	☒	Methods and apparatus for power control in a scalable array of processor elements	713/322
47	US 20020002573 A1	☒	Processor with reconfigurable arithmetic data path	708/501
48	US 20010018733 A1	☒	Array-type processor	712/16
49	US 20010014096 A1	☒	METHOD AND APPARATUS FOR MULTICAST OF ATM CELLS WHERE CONNECTIONS CAN BE DYNAMICALLY ADDED OR DROPPED	370/395.2
50	US 6697372 B1	☒	Local area network accessory for integrating USB connectivity in existing networks	370/402
51	US 6647468 B1	☒	Method and system for optimizing translation buffer recovery after a miss operation within a multi-processor environment	711/147
52	US 6643260 B1	☒	Method and apparatus for implementing a quality of service policy in a data communications network	370/235

Ninth Embodiment

The ninth embodiment of the present invention will be described next.

FIG. **17** is a block diagram showing the construction of an interrupt control apparatus according to the ninth embodiment. In FIG. **17**, the same reference numerals as in FIG. **12** denote the same blocks as in FIG. **12**, respectively, and a detailed description thereof will be omitted.

Referring to FIG. **17**, reference numeral **341'** denotes an instruction word decoder; **435** denotes a normal interrupt return controller; and **436** denotes a break-interrupt return controller.

The instruction word decoder **431'** decodes an instruction word **473** supplied from an instruction fetch section **420**, like the instruction word decoder **431** shown in FIG. **12**. When it is detected by decoding that the supplied instruction word **473** is an instruction word for generating a break-interrupt by a software breakpoint, the instruction word decoder **431'** of this embodiment supplies a break-interrupt generation instruction to a breakpoint controller **433**.

When the supplied instruction word **473** is an instruction word for returning the processor from a normal interrupt state, the instruction word decoder **431'** supplies a normal interrupt return instruction to the normal interrupt return controller **435**. When the supplied instruction world **473** is an instruction word for returning the processor from a break-interrupt state, the instruction word decoder **431'** supplies a break-interrupt return instruction to the break-interrupt return controller **436**. When the supplied instruction word **473** is an instruction word of another type, the instruction word decoder **431'**. supplies the decoded instruction to an instruction execution controller **432**.

The normal interrupt return controller **435** executes a return operation from the normal interrupt state in accordance with the normal interrupt return instruction supplied from the instruction word decoder **431'**. The break-interrupt return controller **436** executes a return operation from the break-interrupt state in accordance with the break-interrupt return instruction supplied from the instruction word decoder **431'**.

More specifically, in the fifth embodiment shown in FIG. **12**, it is specified in accordance with the value of the operand **102** in the interrupt return instruction shown in FIG. **13** whether the operation is a return operation from a normal interrupt state or break-interrupt state. In the ninth embodiment, however, as the contents of an instruction code **101**, two types of return instructions, a return instruction from a normal interrupt state and a return instruction from a break-interrupt state are used. In accordance with which return instruction is supplied to the instruction word decoder **431'**, the interrupt return controller which should perform the return operation is specified, and the processor operation information before the interrupt is restored. In this case, an operand **102** has an arbitrary value.

In this embodiment, a normal return address register **452**, a normal previous state register **453**, and a normal factor register **454** constitute the first information holding section of the present invention, and a break return address register **455** constitutes the second information holding section of the present invention.

In this embodiment as well, the interrupt return instruction is prepared at the end of each of an interrupt handler for a normal interrupt and an interrupt handler for a break-interrupt. The instruction code **101** of an interrupt return instruction for a normal interrupt is constructed by a return instruction from a normal interrupt, and the instruction code **101** of an interrupt return instruction for a break-interrupt is constructed by a return instruction from a break-interrupt. The two types of interrupt return instructions constitute the return operation specifying section of the present invention.

The operation of the interrupt control apparatus of this embodiment will be described next by exemplifying the processor state transition shown in FIG. **7**: normal state **201**→normal interrupt state **202**→break-interrupt state **204**→normal interrupt state **202**→normal state **201**.

In this embodiment, when the processor transits to an interrupt state due to a normal interrupt or break-interrupt, the same operation as that of the interrupt control apparatus shown in FIG. **12** is performed.

Hence, only the return operation from a normal interrupt state and break-interrupt state, i.e., state transition: break-interrupt state **204**→normal interrupt state **202**→normal state **201** will be described below.

When the processor is in the break-interrupt state **204**, and processing by an interrupt handler corresponding to the break-interrupt is ended, the processor executes an interrupt return instruction **216** at the end of the interrupt handler. The instruction code **101** of the interrupt return instruction executed at this time means an interrupt return instruction from the break-interrupt state. For this reason, the instruction word decoder **431'** decodes the interrupt return instruction and consequently supplies a break-interrupt return instruction to the break-interrupt return controller **436**. When receiving the break-interrupt return instruction, the break-interrupt return controller **436** controls the instruction fetch section **420** and the register section **450** to perform processing as follows.

First, the break-interrupt return controller **436** reads out, from the break return address register **455**, an original instruction address **475** to which the processor will return from the break-interrupt state, supplies the instruction address to the instruction fetch section **420**, and sets the address value in a program counter **422**.

On the basis of an original instruction address **471** set in the program counter **422**, an instruction fetch controller **421** reads out an instruction word **472** of the interrupt handler corresponding to the normal interrupt from a memory **410**, temporarily holds the instruction word in an instruction word register **423**, and then supplies the instruction word to an instruction execution section **430**. Thus, the processor returns from the break-interrupt state **204** to the normal interrupt state **202**. The instruction execution section **430** executes the remaining part of processing of the interrupt handler corresponding to the normal interrupt.

When processing of the interrupt handler corresponding to the normal interrupt is ended in the normal interrupt state **202**, the processor executes an interrupt return instruction **212** at the end of the interrupt handler. The instruction code **101** of the interrupt return instruction executed at this time means an interrupt return instruction from the normal interrupt state. For this reason, the instruction word decoder **431'** decodes the interrupt return instruction and consequently supplies a normal interrupt return instruction to the normal interrupt return controller **435**.

When receiving the normal interrupt return instruction, the normal interrupt return controller **435** controls the instruction fetch section **420** and register section **450** to start the normal interrupt return operation. For this normal interrupt return operation, the same operation as that of the interrupt control apparatus shown in FIG. **12** is performed, and the normal interrupt state **202** transits to the normal state **201**.

	Document ID	U	Title	Current OR
53	US 6628999 B1	☒	Single-chip audio system volume control circuitry and methods	700/94
54	US 6618360 B1	☒	Method for testing data path of peripheral server devices	370/248
55	US 6604136 B1	☒	Application programming interfaces and methods enabling a host to interface with a network processor	709/223
56	US 6601157 B1	☒	Register addressing	711/219
57	US 6532531 B1	☒	Method frame storage using multiple memory circuits	712/202
58	US 6523054 B1	☒	Galois field arithmetic processor	708/492
59	US 6496660 B2	☒	Image formation apparatus with printer engine control which judges whether recording sheets can be fed	399/16
60	US 6493831 B1	☒	Timer circuits for a microcomputer	713/502
61	US 6446190 B1	☒	Register file indexing methods and apparatus for providing indirect control of register addressing in a VLIW processor	712/24
62	US 6434221 B1	☒	Digital subscriber line access and network testing multiplexer	379/27.01
63	US 6430684 B1	☒	Processor circuits, systems, and methods with efficient granularity shift and/or merge instruction(s)	712/300
64	US 6405093 B1	☒	Signal amplitude control circuitry and methods	700/94
65	US 6389029 B1	☒	Local area network incorporating universal serial bus protocol	370/402
66	US 6378021 B1	☒	Switch control method and apparatus in a system having a plurality of processors	710/317
67	US 6338105 B1	☒	Data transmission method and game system constructed by using the method	710/72
68	US 6324603 B1	☒	Data transmission system and game system using the same	710/72
69	US 6310879 B1	☒	Method and apparatus for multicast of ATM cells where connections can be dynamically added or dropped	370/397
70	US 6247036 B1	☒	Processor with reconfigurable arithmetic data path	708/603
71	US 6234469 B1	☒	Money processing apparatus and method	271/3.04
72	US 6226758 B1	☒	Sample rate conversion of non-audio AES data channels	713/600
73	US 6213879 B1	☒	Data transmission system and game system with game peripherals using same	463/36
74	US 6145085 A	☒	Method and apparatus for providing remote access to security features on a computer network	713/202
75	US 6131152 A	☒	Planar cache layout and instruction stream therefor	712/24

As described above, in the ninth embodiment, a normal interrupt return instruction whose instruction code **101** means a return instruction from a normal interrupt state is prepared at the end of an interrupt handler corresponding to a normal interrupt, and a break-interrupt return instruction whose instruction code **101** means a return instruction from a break-interrupt state is prepared at the end of an interrupt handler corresponding to a break-interrupt. In returning from an interrupt, the return operation is specified in accordance with the contents of the instruction code **101**.

According to this construction, even within the interrupt inhibition period immediately after interrupt processing by a normal interrupt and immediately before interrupt return, a break-interrupt can occur. In addition, the processor operation information before the interrupt can be restored with a minimum hardware resource. Furthermore, using two types of instructions: a return instruction from a normal interrupt state and a return instruction from a break-interrupt state, a return instruction for an interrupt can be executed, the calculation time can be shortened, and even when a break-interrupt occurs in a normal interrupt state, the processor operation information before the break-interrupt can be quickly and accurately restored.

In the ninth embodiment, only the break return address register **455** is provided as the second information holding section of the present invention. However, not only the break return address register **455** but also one or both of a break previous state register **458** and a break factor register **459** may be provided, like in the sixth to eighth embodiments shown in FIGS. **14** to **16**.

In this case, when the processor transits to an interrupt state due to a normal interrupt or break-interrupt, the same operation as that of the interrupt control apparatuses shown in FIGS. **14** to **16** is performed. As for the return operation from the interrupt state, the return operation from a normal interrupt state is executed by the normal interrupt return controller **435**, and the return operation from a break-interrupt state is executed by the break-interrupt return controller **436**. Control of the registers is the same as that in the interrupt control apparatuses shown in FIGS. **14** to **16**.

The above embodiments can be applied to debug an interrupt handler corresponding to exceptional processing or external interrupt, and also to debug an interrupt handler corresponding to system call or supervisor call of an OS (Operating System).

10th Embodiment

The 10th embodiment of the present invention will be described below with reference to drawings.

FIG. **18** is a block diagram showing the construction of a data processing system (processor) according to the 10th embodiment for implementing an instruction break scheme by a hardware mechanism.

In FIG. **18**, the same reference numerals as in FIG. **3** denote the same functional parts as in FIG. **3**, respectively, and a detailed description thereof will be omitted.

In the 10th embodiment, an instruction break detection section **23** has, in place of the determination sections 25_{-0} to 25_{-n} used in FIG. **3**, determination sections 100_{-0} to 100_{-n} for performing determination processing different from that of the determination sections 25_{-0} to 25_{-n}.

These determination sections 100_{-0} to 100_{-n} receive not only instruction break addresses and flags held in breakpoint registers 24_{-0} to 24_{-n} prepared in units of determination sections 100_{-0} to 100_{-n} and current execution address supplied from a program counter **21** but also the condition code of a conditional instruction, which is read out from a condition register **51** in a register section **50**, and an instruction word that is read out from an instruction register **22** and currently being executed.

FIG. **19** is a block diagram showing the construction of the determination section 100_{-0} as a representative of the determination sections 100_{-0} to 100_{-0}. Each of the remaining determination sections 100_{-1} to 100_{-n} has the same construction as in FIG. **19**.

As shown in FIG. **19**, the determination section 100_{-0} of this embodiment comprises a comparison section **101**, a conditional instruction decoder **102**, a condition determination section **103**, and an AND circuit **104**.

The comparison section **101** compares an instruction break address held in an address register **24***a* of the breakpoint register 24_{-0} provided in accordance with the determination section 100_{-0} with a current execution address supplied from the program counter **21** and determines whether the two addresses match. When the two addresses match, the comparison section **101** outputs a determination signal having the value "1". When the two addresses do not match, the comparison section **101** outputs a determination signal having the value "0".

The conditional instruction decoder **102** decodes the instruction word that is supplied from the instruction register **22** and currently being executed to detect whether the instruction is a conditional instruction, and supplies the detection result to the condition determination section **103**. On the basis of the decoding result of the instruction word, which is supplied from the conditional instruction decoder **102**, the condition determination section **103** determines whether the condition designated by the condition code of the conditional instruction, which is supplied from the condition register **51**, is satisfied. If the condition designated by the condition code of the conditional instruction is satisfied, a determination signal having the value "1" is output. If the condition is not satisfied, a determination signal having the value "0" is output. If it is found by decoding by the conditional instruction decoder **102** that the supplied instruction is not a conditional instruction, the condition determination section **103** outputs a determination signal having the value "0".

The AND circuit **104** performs AND operation to the value of a flag register **24***b* of the breakpoint register 24_{-0} provided in accordance with the determination section 100_{-0} the determination signal output from the comparison section **101** and related to the instruction break address, and the determination signal output from the condition determination section **103** and related to the condition code, and outputs the AND operation result to an OR circuit **26** shown in FIG. **18**.

According to this construction, in at least one entry of the determination 'sections 100_{-0} to 100_{-n} of the instruction break detection section **23**, when the instruction break address and the current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied, the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

As described above, in the 10th embodiment, each of the determination sections 100_{-0} to 100_{-n} determines not only whether the instruction break generation condition for the instruction break address and the flag value is satisfied but also whether the condition of the conditional instruction is satisfied. Only when both conditions are satisfied, a break-interrupt occurs.

	Document ID	U	Title	Current OR
76	US 6122748 A	☒	Control of computer system wake/sleep transitions	713/323
77	US 6098915 A	☒	Tape winding apparatus and tape winding method	242/527
78	US 6094730 A	☒	Hardware-assisted firmware tracing method and apparatus	714/28
79	US 6085275 A	☒	Data processing system and method thereof	710/316
80	US 6079188 A	☒	Packaging container production equipment and packaging container production method	53/451
81	US 6067778 A	☒	Packaging container production equipment and packaging container production method	53/451
82	US 6052757 A	☒	Content addressable memory FIFO with and without purging	711/108
83	US 6044225 A	☒	Multiple parallel digital data stream channel controller	710/52
84	US 6038643 A	☒	Stack management unit and method for a processor having a stack	711/132
85	US 6009499 A	☒	Pipelined stack caching circuit	711/132
86	US 6000051 A	☒	Method and apparatus for high-speed interconnect testing	714/727
87	US 5999737 A	☒	Link time optimization via dead code elimination, code motion, code partitioning, code grouping, loop analysis with code motion, loop invariant analysis and active variable to register analysis	717/162
88	US 5986426 A	☒	Adaptive pulse width modulated motor control	318/599
89	US 5979755 A	☒	Auto-changer	235/383
90	US 5974491 A	☒	High speed data transfer apparatus for duplexing system	710/106
91	US 5966539 A	☒	Link time optimization with translation to intermediate program and following optimization techniques including program analysis code motion live variable set generation order analysis, dead code elimination and load invariant analysis	717/156
92	US 5909463 A	☒	Single-chip software configurable transceiver for asymmetric communication system	375/220
93	US 5903718 A	☒	Remote program monitor method and system using a system-under-test microcontroller for self-debug	714/38
94	US 5894549 A	☒	System and method for fault detection in microcontroller program memory	714/42
95	US 5892738 A	☒	Disk recording-playback device and disk loading or unloading method	369/30.32
96	US 5878274 A	☒	Intelligent multi modal communications apparatus utilizing predetermined rules to choose optimal combinations of input and output formats	710/8
97	US 5862398 A	☒	Compiler generating swizzled instructions usable in a simplified cache layout	712/24

Thus, in debugging a program including a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the conditional instruction is satisfied, a break-interrupt occurs. When the condition of the conditional instruction is not satisfied, or the supplied instruction is an unconditional instruction, a break-interrupt can be inhibited.

11th Embodiment

The 11th embodiment of the present invention will be described next with reference to drawings.

FIG. **20** is a block diagram showing the construction of a data processing system (processor) according to the 11th embodiment for implementing an instruction break scheme by a hardware mechanism.

In FIG. **20**, the same reference numerals as in FIG. **18** denote the same functional parts as in FIG. **18**, respectively, and a detailed description thereof will be omitted.

In the 10th embodiment shown in FIG. **18**, a scalar processor for executing one unit of processing in accordance with one instruction has been described. The 11th embodiment shown in FIG. **20** is applied to a VLIW (Very Long Instruction Word) type processor which designates processing operations by one instruction and executes those operations in parallel.

More specifically, as shown in FIG. **20**, an instruction register **27** of the 11th embodiment is designed to hold a fixed-length long instruction word formed from short instructions. Each of determination sections $\mathbf{100}_0$ to $\mathbf{100}_n$ of an instruction break detection section **23** has a construction shown in FIG. **21**. FIG. **21** is a block diagram showing the construction of the determination section $\mathbf{100}_n$ as a representative of the determination sections $\mathbf{100}_0$ to $\mathbf{100}_n$. In FIG. **21**, the same reference numerals as in FIG. **19** denote the same blocks as in FIG. **19**, respectively, and a detailed description thereof will be omitted.

As shown in FIG. **21**, the determination section $\mathbf{100}_0$ of this embodiment comprises a comparison section **101**, a conditional instruction decoder **111**, a condition determination section **112**, an OR circuit **113**, and an AND circuit **104**.

The conditional instruction decoder **111** has conditional instruction decoders $\mathbf{111}_0$ and $\mathbf{111}_i$ provided in accordance with short instructions IR#0 to IR#i forming the long instruction word held in the instruction register **27** to decode each short instruction supplied from the instruction register **27** and detect whether the instruction is a conditional instruction.

The condition determination section **112** has condition determination sections $\mathbf{112}_0$ to $\mathbf{112}_i$ provided in accordance with the conditional instruction decoders $\mathbf{111}_n$ and $\mathbf{111}_i$, respectively. Each of the condition determination sections $\mathbf{112}_0$ to $\mathbf{112}_i$ determines, on the basis of the short instruction decoding result ,supplied from a corresponding one of the conditional instruction decoders $\mathbf{111}_0$ and $\mathbf{111}_i$, whether the condition designated by the condition code of the conditional instruction supplied from a condition register **51** is satisfied. If the condition of the conditional instruction is satisfied, a determination signal having the value "1" is output. If the condition is not satisfied, a determination signal having the value "0" is output. If it is found by decoding by the conditional instruction decoders $\mathbf{111}_0$ to $\mathbf{111}_i$ that the supplied instructions are not conditional instructions, the condition determination sections $\mathbf{112}_0$ to $\mathbf{112}_i$ output determination signals each having the value "0".

The OR circuit **113** performs OR operation to the determination signals output from the condition determination sections $\mathbf{112}_0$ to $\mathbf{112}_i$ and outputs the OR operation result to the AND circuit **104**. The AND circuit **104** performs AND operation to the value of a flag register ***24b*** of a breakpoint register $\mathbf{24}_0$ provided in accordance with the determination section $\mathbf{100}_0$, the determination signal output from the comparison section **101** and related to the instruction break address, and the determination signal output from the OR circuit **113** and related to the condition code, and outputs the AND operation result to an OR circuit **26** shown in FIG. **20**.

According to this construction, in at least one entry of the determination sections $\mathbf{100}_0$ to $\mathbf{100}_n$ of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register ***24b*** is "1", and the condition of the conditional instruction is satisfied for at least one of the short instructions forming the long instruction word, the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

As described above, in the 11th embodiment, each of the determination sections $\mathbf{100}_0$ to $\mathbf{100}_n$ determines not only whether the instruction break generation condition for the instruction break address and the flag value is satisfied but also whether the condition of the conditional instruction is satisfied for each of the short instructions forming the long instruction word. Only when both conditions are satisfied, a break-interrupt occurs.

Thus, when the short instructions forming the long instruction word include a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the conditional instruction is satisfied for any one of the short instructions, a break-interrupt occurs. When the condition of the conditional instruction is not satisfied for none of the short instructions, or all the short instructions are unconditional instructions, a break-interrupt can be inhibited.

12th Embodiment

The 12th embodiment of the present invention will be described next with reference to drawings.

FIG. **22** is a block diagram showing the construction of a data processing system (processor) according to the 12th embodiment for implementing an instruction break scheme by a hardware mechanism.

The processor according to the 12th embodiment shown in FIG. **22** is also a VLIW type processor, like the processor according to the 11th embodiment shown in FIG. **20**. In FIG. **22**, the same reference numerals as in FIG. **20** denote the same blocks as in FIG. **20**, respectively.

In the 12th embodiment shown in FIG. **22**, each of breakpoint registers $\mathbf{24}_0$ to $\mathbf{24}_n$ of an instruction break detection section **23** has a displacement register ***24c*** for holding displacement information from the start portion of a long instruction word as a breakpoint target, in addition to an address register ***24a*** for holding the target address of a breakpoint at which execution is to be stopped and a flag register ***24b*** indicating whether an instruction break operation is valid. The displacement information held in the displacement register ***24c*** is used together with the instruction break address held in the address register ***24a***, thereby specifying one of short instructions forming one long instruction word.

Each of determination sections $\mathbf{120}_n$ to $\mathbf{120}_n$ according to the 12th embodiment has a construction shown in FIG. **23**.

	Document ID	U	Title	Current OR
98	US 5848255 A	☒	Method and aparatus for increasing the number of instructions capable of being used in a parallel processor by providing programmable operation decorders	712/212
99	US 5841766 A	☒	Diversity-oriented channel allocation in a mobile communications system	370/321
100	US 5822553 A	☒	Multiple parallel digital data stream channel controller architecture	710/305
101	US 5816527 A	☒	Tape winding apparatus and tape winding method	242/527
102	US 5805874 A	☒	Method and apparatus for performing a vector skip instruction in a data processor	712/222
103	US 5799167 A	☒	Instruction nullification system and method for a processor that executes instructions out of order	712/218
104	US 5797043 A	☒	System for managing the transfer of data between FIFOs within pool memory and peripherals being programmable with identifications of the FIFOs	710/56
105	US 5794082 A	☒	Electronic flash device with slave emission function	396/56
106	US 5784649 A	☒	Multi-threaded FIFO pool buffer and bus transfer control system	710/52
107	US 5779181 A	☒	Tape winding apparatus and tape winding method	242/532.6
108	US 5771363 A	☒	Single-chip microcomputer having an expandable address area	712/200
109	US 5754805 A	☒	Instruction in a data processing system utilizing extension bits and method therefor	712/200
110	US 5752074 A	☒	Data processing system and method thereof	712/29
111	US 5745721 A	☒	Partitioned addressing apparatus for vector/scalar registers	712/208
112	US 5742786 A	☒	Method and apparatus for storing vector data in multiple non-consecutive locations in a data processor using a mask value	711/217
113	US 5740378 A	☒	Hot swap bus architecture	710/302
114	US 5737586 A	☒	Data processing system and method thereof	712/236
115	US 5734879 A	☒	Saturation instruction in a data processor	712/221
116	US 5717947 A	☒	Data processing system and method thereof	712/3
117	US 5706488 A	☒	Data processing system and method thereof	712/223
118	US 5706439 A	☒	Method and system for matching packet size for efficient transmission over a serial bus	370/234
119	US 5687344 A	☒	Single-chip microcomputer having an expandable address area	711/220
120	US 5664134 A	☒	Data processor for performing a comparison instruction using selective enablement and wired boolean logic	712/245

FIG. **23** is a block diagram showing the construction of the determination section 120_{-0} as a representative of the determination sections 120_{-0} to 120_{-n}. In FIG. **23**, the same reference numerals as in FIGS. **19** and **21** denote the same blocks as in FIGS. **19** and **21**, respectively, and a detailed description thereof will be omitted.

As shown in FIG. **23**, the determination section 120_{-0} of this embodiment comprises a comparison section **101**, a conditional instruction decoder **102**, a condition determination section **103**, an AND circuit **104**, and a selector **121**.

The selector **121** switches the selection state using, as a control signal, displacement information held in the displacement register **24***c* of the breakpoint register 24_{-0} provided in accordance with the determination section 120_{-0} whereby only a short instruction designated by the displacement information from short instructions IR#**0** to IR#i is selectively output to the conditional instruction decoder **102**. The conditional instruction decoder **102** decodes the short instruction selected by the selector **121** to detect whether the instruction is a conditional instruction, and supplies the decoding result to the condition determination section **103**.

According to this construction, in at least one entry of the determination sections 120_{-0} to 120_{-n} of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied for a selected instruction of the short instructions forming the long instruction word, an OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

As described above, in the 12th embodiment, each of the determination sections 120_{-0} to 120_{-n} determines not only whether the instruction break generation condition for the instruction break address and the flag value is satisfied but also whether the condition of the conditional instruction specified by the displacement information is satisfied for each of the short instructions forming the long instruction word. Only when both conditions are satisfied, a break-interrupt occurs.

Thus, when a specific instruction selected by the displacement information from the short instructions forming the long instruction word is a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the specified conditional instruction is satisfied, a break-interrupt occurs. When the condition of the specified conditional instruction is not satisfied, or the specified short instruction is an unconditional instruction, a break-interrupt can be inhibited.

13th Embodiment

The 13th embodiment of the present invention will be described next with reference to drawings.

FIG. **24** is a block diagram showing the construction of a data processing system (processor) according to the 13th embodiment for implementing an instruction break scheme by a hardware mechanism.

In FIG. **24**, the same reference numerals as in FIGS. **18** and **20** denote the same functional parts as in FIGS. **18** and **20**, respectively, and a detailed description thereof will be omitted.

In the 10th embodiment shown in FIG. **18**, a scalar processor for executing one unit of processing in accordance with one instruction has been described. In the 11th embodiment shown in FIG. **20**, a VLIW type processor which parallelly executes short instructions forming a fixed-length long instruction word has been described. The 13th embodiment shown in FIG. **24** is applied to a parallel processor for executing one basic instruction or, in parallel, basic instructions forming a variable-length instruction word.

More specifically, as shown in FIG. **24**, an instruction register **28** of the 13th embodiment is designed to hold a variable-length instruction word formed from one or more basic instructions. Each of determination sections 130_{-0} to 130_{-n} of an instruction break detection section **23** has a construction shown in FIG. **25**. FIG. **25** is a block diagram showing the construction of the determination section 130_{-n} as a representative of the determination sections 130_{-n} to

As shown in FIG. **25**, the determination section 130_{-0} of this embodiment comprises a comparison section **101**, a valid instruction encoder **131**, a conditional instruction decoder **111**, a condition determination section **112**, an AND circuit **132**, an OR circuit **113**, and another AND circuit **104**. In FIG. **25**, the same reference numerals as in FIG. **21** denote the same functional parts as in FIG. **21**, respectively, and a detailed description thereof will be omitted.

In the VLIW type processor shown in FIG. **21**, one long instruction word has a fixed length, and the number of short instructions forming it is constant. Hence, the instruction register **27** always stores i short instructions IR#**0** to IR#i (an unexecuted instruction such as "nop" is stored at an unused portion).

To the contrary, in the parallel processor according to the 13th embodiment shown in FIG. **25**, the instruction word stored in the instruction register **28** has a variable length, and the number of basic instructions forming it is variable. Hence, the instruction register **28** stores an arbitrary number of basic instructions and i basic instructions at maximum from the left side (sequentially from IR#**0** side). In this case, each basic instruction has, at its head portion, a one-bit flag representing whether the basic instruction is at the end of a variable-length instruction word.

The valid instruction encoder **131** encodes the basic instructions stored in the instruction register **28** and refers to the flag information described at the head portion of each basic instruction, thereby detecting the number of valid basic instructions stored in the instruction register **28**. An encoded signal having the value "1" is output sequentially from the IR#**0** side in number, equal to the number of basic instructions stored in the instruction register **28**, and an encoded signal having the value "0" is output for the remaining portions.

The AND circuit **132** includes AND circuits 132_{-0} to 132_{-i} provided in accordance with determination sections 112_{-0} to 112_{-i}, respectively. Each of the AND circuits 132_{-0} to 132_{-i} performs AND operation to a determination signal representing whether the condition of a conditional instruction obtained by a corresponding one of the determination sections 112_{-0} to 112_{-i} and an encoded signal output from the valid instruction encoder **131** in accordance with each of the basic instructions IR#**0** to IR#i in the instruction register **28**, and outputs the AND operation result to the OR circuit **113**.

The OR circuit **113** performs OR operation to the signals output from the condition determination sections 132_{-0} to 132_{-i} and outputs the OR operation result to the AND circuit **104**. The AND circuit **104** performs AND operation to the value of a flag register **24***b* of a breakpoint register 24_{-0} provided in accordance with the determination section 130_{-0}, the determination signal output from the comparison

	Document ID	U	Title	Current OR
121	US 5659706 A	☒	Vector/scalar processor with simultaneous processing and instruction cache filling	711/125
122	US 5657288 A	☒	Efficient addressing of large memories	365/230.02
123	US 5652421 A	☒	Method and apparatus for generating gift certificates	235/381
124	US 5640524 A	☒	Method and apparatus for chaining vector instructions	712/222
125	US 5623650 A	☒	Method of processing a sequence of conditional vector IF statements	712/234
126	US 5600846 A	☒	Data processing system and method thereof	712/5
127	US 5598571 A	☒	Data processor for conditionally modifying extension bits in response to data processing instruction execution	712/9
128	US 5598547 A	☒	Vector processor having functional unit paths of differing pipeline lengths	712/222
129	US 5592257 A	☒	Electronic flash device with slave emission function	396/171
130	US 5585987 A	☒	Camera system including electronic flash device with slave emission function	396/171
131	US 5572689 A	☒	Data processing system and method thereof	712/200
132	US 5559973 A	☒	Data processing system and method thereof	712/241
133	US 5553287 A	☒	Methods and apparatus for dynamically using floating master interlock	709/201
134	US 5548768 A	☒	Data processing system and method thereof	712/200
135	US 5544337 A	☒	Vector processor having registers for control by vector resisters	712/4
136	US 5541933 A	☒	Auto-detection of DDS service and line rate	714/708
137	US 5537562 A	☒	Data processing system and method thereof	712/234
138	US 5530901 A	☒	Data Transmission processing system having DMA channels running cyclically to execute data transmission from host to memory and from memory to processing unit successively	710/28
139	US 5513368 A	☒	Computer I/O adapters for programmably varying states of peripheral devices without interfering with central processor operations	710/22
140	US 5500514 A	☒	Method and apparatus for generating gift certificates	235/381
141	US 5430884 A	☒	Scalar/vector processor	712/3
142	US 5426744 A	☒	Single chip microprocessor for satisfying requirement specification of users	712/228
143	US 5377324 A	☒	Exclusive shared storage control system in computer system	711/148

section **101** and related to the instruction break address, and the determination signal output from the OR circuit **113** and related to the condition code, and outputs the AND operation result to an OR circuit **26** shown in FIG. **24**.

According to this construction, in at least one entry of the determination sections 130_0 to 130_n of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied for at least one of the basic instructions forming the variable-length instruction word, the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

As described above, in the 13th embodiment, each of the determination sections 130_0 to 130_n determines not only whether the instruction break generation condition for the instruction break address and the flag value is satisfied but also whether the condition of the conditional instruction is satisfied for each of the basic instructions forming the variable-length instruction word. Only when both conditions are satisfied, a break-interrupt occurs.

Thus, when the basic instructions forming the variable-length instruction word include a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation here the instruction break generation condition is satisfied, when the condition of the conditional instruction is satisfied for any one of the basic instructions, a break-interrupt occurs. When the condition of the conditional instruction is not satisfied for none of the basic instructions, or all the basic instructions are unconditional instructions, a break-interrupt can be inhibited.

14th Embodiment

The 14th embodiment of the present invention will be described next with reference to drawings.

FIG. **26** is a block diagram showing the construction of a data processing system (processor) according to the 14th embodiment for implementing an instruction break scheme by a hardware mechanism.

The processor according to the 14th embodiment shown in FIG. **26** is also a parallel processor, like the processor according to the 13th embodiment shown in FIG. **24**. In FIG. **26**, the same reference numerals as in FIG. **24** denote the same blocks as in FIG. **24**, respectively.

In the 14th embodiment shown in FIG. **26**, each of breakpoint registers 24_0 to 24_n of an instruction break detection section **23** has an address register **24***a*, a flag register **24***b*, and a displacement register **24***c*, like in the 12th embodiment shown in FIG. **22**. Displacement information held in the displacement register **24***c* is used together with the instruction break address held in the address register **24***a*, thereby specifying one of basic instructions forming one variable-length instruction word.

Each of determination sections 140_0 to 140_n according to the 14th embodiment has a construction shown in FIG. **27**. FIG. **27** is a block diagram showing the construction of the determination section 140_0 as a representative of the determination sections 140_0 to 140_n. In FIG. **27**, the same reference numerals as in FIGS. **23** and **25** denote the same blocks as in FIGS. **23** and **25**, respectively, and a detailed description thereof will be Omitted.

As shown in FIG. **27**, the determination section 140_0 of this embodiment comprises a comparison section **101**, a conditional instruction decoder **102**, a condition determination section **103**, an AND circuit **104**, a first selector **121**, a valid instruction encoder **131**, a second selector **141**, and another AND circuit **142**.

The second selector **141** switches the selection state using, as a control signal, displacement information held in the displacement register **24***c* of the breakpoint register 24_0 provided in accordance with the determination section 140_0 whereby only an encoded signal designated by the displacement information from encoded signals output from the valid instruction encoder **131** in accordance with basic instructions IR#0 to IR#i in an instruction register **28** is selectively output to the AND circuit **142**.

The AND circuit **142** performs AND operation to the condition determination signal obtained by the condition determination section **103** for the conditional instruction selected by the first selector **121** on the basis of,the displacement information, and the encoded signal selected by the second selector **141** on the basis of the displacement information, and outputs the AND operation result to the AND circuit **104**.

According to this construction, in at least one entry of the determination sections 140_0 to 140_n of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied for a selected instruction of the basic instructions forming the variable-length instruction word, an OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

As described above, in the 14th embodiment, each of the determination sections 140_0 to 140_n determines not only whether the instruction break generation condition for the instruction break address and flag value is satisfied but also whether the condition of the conditional instruction specified by the displacement information is satisfied for each of the basic instructions forming the variable-length instruction word of the parallel processor. Only when both conditions are satisfied, a break-interrupt occurs.

Thus, when an instruction selected by the displacement information from the basic instructions forming the variable-length instruction word is a conditional instruction, a break-interrupt can be controlled in accordance with whether the condition of the conditional instruction is satisfied. More specifically, in a situation where the instruction break generation condition is satisfied, when the condition of the specified conditional instruction is satisfied, a break-interrupt occurs. When the condition of the specified conditional instruction is not satisfied, or the specified basic instruction is an unconditional instruction, a break-interrupt can be inhibited.

15th Embodiment

The 15th embodiment of the present invention will be described next with reference to drawings.

FIG. **28** is a block diagram showing the construction of a data processing system (scalar processor) according to the 15th embodiment for implementing an instruction break scheme by a hardware mechanism. In FIG. **28**, the same reference numerals as in FIG. **18** denote the same functional parts as in FIG. **18**, respectively, and a detailed description thereof will be omitted.

In the 15th embodiment shown in FIG. **28**, each of breakpoint registers 24_0 to 24_n of an instruction break detection section **23** has a mode register **24***d* for holding mode information on an instruction break mode or condi-

	Document ID	U	Title	Current OR
144	US 5331667 A	☒	Telephone exchange apparatus with communication line clocking	375/356
145	US 5327428 A	☒	Collision-free insertion and removal of circuit-switched channels in a packet-switched transmission structure	370/353
146	US 5297262 A	☒	Methods and apparatus for dynamically managing input/output (I/O) connectivity	710/36
147	US 5297239 A	☒	Compile type knowledge processing tool, a high-speed inference method therefor and a system using the tool	706/59
148	US 5291197 A	☒	One-chip data processor with built-in A/D converter for automatically repeating A/D conversions without instructions from a CPU	341/141
149	US 5280593 A	☒	Computer system permitting switching between architected and interpretation instructions in a pipeline by enabling pipeline drain	712/208
150	US 5273719 A	☒	Urine treating device	422/122
151	US 5263020 A	☒	Echo canceller	370/289
152	US 5226164 A	☒	Millicode register management and pipeline reset	712/209
153	US 5189636 A	☒	Dual mode combining circuitry	708/706
154	US 5149399 A	☒	Liquid evaporator	159/22
155	US 5121424 A	☒	Telephone system and speech level adjusting method therefor	379/165
156	US 5047975 A	☒	Dual mode adder circuitry with overflow detection and substitution enabled for a particular mode	708/706
157	US 5043895 A	☒	Method of controlling gear changing operation in automatic transmission	701/66
158	US 5014193 A	☒	Dynamically configurable portable computer system	710/10
159	US 4970718 A	☒	Apparatus for supplying channel-control signals and maintenance signals in a serial data concentrator system	370/434
160	US 4916738 A	☒	Remote access terminal security	713/159
161	US 4903296 A	☒	Implementing a shared higher level of privilege on personal computers for copy protection of software	705/56
162	US 4881194 A	☒	Stored-program controller for equalizing conditional branch delays	712/233
163	US 4841828 A	☒	Electronic musical instrument with digital filter	84/601
164	US 4835607 A	☒	Method and apparatus for expanding compressed video data	348/390.1
165	US 4827433 A	☒	Processing device for changing magnification of image data	345/668
166	US 4823201 A	☒	Processor for expanding a compressed video signal	375/240.08

tional instruction break mode, in addition to an address register **24***a* for holding the target address of a breakpoint at which execution is to be stopped and a flag register **24***b* indicating whether an instruction break operation is valid. The mode register **24***d* having the value "0" means an instruction break mode, and the mode register **24***d* having the value "1" means a conditional instruction break mode.

In the instruction break mode, when instruction break generation condition held by the address register **24***a* and the flag register **24***b* are satisfied, a break-interrupt occurs. In the conditional instruction break mode, a break-interrupt occurs when not only the instruction break, generation conditions but also the condition of a conditional instruction is satisfied, as described in the 10th embodiment.

Each of determination sections $\mathbf{150}_{-0}$ to $\mathbf{150}_{-n}$ according to the 15th embodiment has a construction shown in FIG. **29**. FIG. **29** is a block diagram showing the construction of the determination section $\mathbf{150}_{-0}$ as a representative of the determination sections $\mathbf{150}_{-0}$ to $\mathbf{150}_{-n}$. In FIG. **29**, the same reference numerals as in FIG. **19** denote the same blocks as in FIG. **19**, respectively, and a detailed description thereof will be omitted.

As shown in FIG. **29**, the determination section $\mathbf{150}_{-0}$ of this embodiment comprises a comparison section **101**, a conditional instruction decoder **102**, a condition determination section **103**, two AND circuits **151** and **152**, and an OR circuit **153**.

One AND circuit **151** performs AND operation to a determination signal output from the comparison section **101** and related to an instruction break address, a determination signal output from the condition determination section **103** and related to a condition code, the value of the flag register **24***b* of the breakpoint register $\mathbf{24}_{-0}$ provided in accordance with the determination section $\mathbf{150}_{-0}$, and the value of the mode register **24***d*, and outputs the AND operation result to the OR circuit **153**.

The other AND circuit **152** performs AND operation to the determination signal output from the comparison section **101** and related to an instruction break address, the value of the flag register **24***b* of the breakpoint register $\mathbf{24}_{-0}$ provided in accordance with the determination section $\mathbf{150}_{-0}$, and a value obtained by inverting the value of the mode register **24***d*, and outputs the AND operation result to the OR circuit **153**. The OR circuit **153** performs OR operation to the signals output from the two AND circuits **151** and **152**, and outputs the OR operation result to an OR circuit **26** shown in FIG. **28**.

According to this construction, in a situation where the conditional instruction break mode is designated by mode information stored in the mode register **24***d*, in at least one entry of the determination sections $\mathbf{150}_{-0}$ to $\mathbf{150}_{-n}$ of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied, the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

On the other hand, in a situation where the instruction break mode is designated by mode information stored in the mode register **24***d*, in at **129** least one entry of the determination sections $\mathbf{150}_{-0}$ to $\mathbf{150}_{-n}$ of the instruction break detection section **23**, when the instruction break address and current execution address match, and the value of the flag register **24***b* is "1", the OR circuit **26** outputs the interrupt notification signal **67** to the interrupt control section **40**.

As described above, according to the 15th embodiment, when the conditional instruction break mode is designated, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. In addition, when the instruction break mode is designated, a break-interrupt can be controlled in accordance with whether the instruction break generation condition is satisfied regardless of whether the condition of the conditional instruction is satisfied.

In the 15th embodiment, an construction in which the mode register **24***d* is added to the scalar processor described in the 10th embodiment, and accordingly, the two AND circuits **151** and **152** and the OR circuit **153** are provided in each of the determination sections $\mathbf{150}_{-0}$ to $\mathbf{150}_{-n}$ has been described. These components may be added to the VLIW type processor and parallel processor described in the 11th to 14th embodiments.

In this case, the determination sections have constructions shown in FIGS. **30** to **33**, respectively. In FIGS. **30** to **33**, the same reference numerals as in FIGS. **21**, **23**, **25**, **27**, and **29** denote respectively the same parts as in FIGS. **21**, **23**, **25**, **27**, and **29**, which perform the same operations as described above, and a detailed description thereof will be omitted.

16th Embodiment

The 16th embodiment of the present invention will be described next with reference to drawings.

FIG. **34** is a block diagram showing the construction of a data processing system (scalar processor) according to the 16th embodiment for implementing an instruction break scheme by a hardware mechanism. In FIG. **34**, the same reference numerals as in FIG. **18** denote the same functional parts as in FIG. **18**, respectively, and a detailed description thereof will be omitted.

The 16th embodiment shown in FIG. **34** is different from the 10th embodiment shown in FIG. **18** in the construction of each of determination sections of an instruction break detection section **23**. Each of determination sections $\mathbf{160}_{-0}$ to $\mathbf{160}_{-n}$ of this embodiment has a construction shown in FIG. **35**. FIG. **35** is a block diagram showing the construction of the determination section $\mathbf{160}_{-n}$ as a representative of the determination sections $\mathbf{160}_{-n}$ to $\mathbf{160}_{-n}$. In FIG. **35**, the same reference numerals as in FIG. **19** denote the same blocks as in FIG. **19**, respectively, and a detailed description thereof will be omitted.

As shown in FIG. **35**, the determination section $\mathbf{160}_{-0}$ of this embodiment comprises a comparison section **101**, a conditional instruction decoder **102**, a condition determination section **103**, an AND circuit **104**, an unconditional instruction decoder **161**, and an OR circuit **162**.

The unconditional instruction decoder **161** decodes an instruction word that is supplied from an instruction register **22** and currently being executed to detect whether the instruction word is an unconditional instruction and supplies the decoding result to the OR circuit **162**. When the supplied instruction word is an unconditional instruction, the unconditional instruction decoder **161** output a signal having the value "1".

On the basis of the decoding result from the conditional instruction decoder **102**, which represents whether the instruction is a conditional instruction, the condition determination section **103** of this embodiment determines whether the condition designated by the condition code of the conditional instruction, which is supplied from a condition register **51**, is satisfied, and supplies the determination signal to the OR circuit **162**.

The OR circuit **162** performs OR operation to the condition determination siginal obtained by the condition deter-

	Document ID	U	Title	Current OR
167	US 4817140 A	☒	Software protection system using a single-key cryptosystem, a hardware-based authorization system and a secure coprocessor	705/55
168	US 4816913 A	☒	Pixel interpolation circuitry as for a video signal processor	375/240.08
169	US 4766590 A	☒	Loop transmission system having plural stations connected in a variable order	370/407
170	US 4766566 A	☒	Performance enhancement scheme for a RISC type VLSI processor using dual execution units for parallel instruction processing	712/23
171	US 4760518 A	☒	Bi-directional databus system for supporting superposition of vector and scalar operations in a computer	710/107
172	US 4750103 A	☒	System and method for detecting and controlling knocking in an internal combustion engine	701/111
173	US 4686638 A	☒	Leakage inspection method with object type compensation	702/51
174	US 4682144 A	☒	Light transmission system for trains	246/166.1
175	US 4661974 A	☒	Automatic route selection of a private telephone network path on the basis of a public telephone network number	379/198
176	US 4657443 A	☒	Arrangement for supervising synchronous displacement of the pistons of two cylinder-and-piston units	405/302
177	US 4656658 A	☒	Network routing arrangement	379/221.06
178	US 4644393 A	☒	Implementing a shared higher level of privilege on personal computers for copy protection of software	705/56
179	US 4643048 A	☒	Method of controlling automatic transmission in accordance with determination of optimum gear ratio	477/124
180	US 4631364 A	☒	Communication system having dynamically assigned station set buttons	379/164
181	US 4548092 A	☒	Bicycle gear shift unit	74/473.14
182	US 4504961 A	☒	Plural-sheet detector	377/8
183	US 4496227 A	☒	Photographic operation control circuit for camera	396/292
184	US 4486626 A	☒	Method of and system for limiting access to a group of telephone trunks	379/196
185	US 4468528 A	☒	Methods and apparatus for providing enhanced announcements in a telephone system	379/84
186	US 4447873 A	☒	Input-output buffers for a digital signal processing system	710/53
187	US 4447786 A	☒	Waveform synthesizer and motor controller	318/811
188	US 4415773 A	☒	Methods of establishing a switching connection within a switching system	379/287
189	US 4400773 A	☒	Independent handling of I/O interrupt requests and associated status information transfers	710/19

mination section **103** for the **132** conditional instruction and the unconditional instruction determination signal obtained by the unconditional instruction decoder **161**, and outputs the OR operation result to the AND circuit **104**.

The AND circuit **104** performs AND operation to the value of a flag register **24***b* of a breakpoint register 24_0 provided in accordance with the determination section 160_0, the determination signal output from the comparison section **101** and related to the instruction break address, and the signal output from the OR circuit **162**, and outputs the AND operation result to an OR circuit **26** shown in FIG. **34**.

According to this construction, in a case wherein the instruction word stored in the instruction register **22** is a conditional instruction, in at least one entry of the determination sections 160_0 to 160_n of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied, the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

On the other hand, when the instruction word stored in the instruction register **22** is an unconditional instruction, the OR circuit **162** outputs a signal having the value "1", like in the case wherein the instruction word is a conditional instruction, and the condition is satisfied. Hence, in at least one entry, when the instruction break address and current execution address match, and the value of the flag register **24***b* is "1", the OR circuit **26** outputs the interrupt notification signal **67** to the interrupt control section **40**.

As described above, according to the 16th embodiment, when the supplied instruction word is a conditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. When the supplied instruction word is an unconditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition is satisfied.

In the 16th embodiment, an construction in which, for the scalar processor described in the 10th embodiment, the unconditional instruction decoder **161** and OR circuit **162** are added in each of the determination sections 160_0 to 160_n has been described. These components may be added to the VLIW type processor and parallel processor described in the 11th to 14th embodiments.

In this case, the determination sections have constructions shown in FIGS. **36** to **39**, respectively.

In FIGS. **36** to **39**, the same reference numerals as in FIGS. **21**, **23**, **25**, **27**, and **35** denote respectively the same parts as in FIGS. **21**, **23**, **25**, **27**, and **35**, which perform the same operations as described above. Each component with a symbol "'" has the same function as that of a corresponding component without this symbol, and its operation will be apparent without a detailed description of FIGS. **36** to **39**.

17th Embodiment

The 17th embodiment of the present invention will be described next with reference to drawings.

FIG. **40** is a block diagram showing the construction of a data processing system (scalar processor) according to the 17th embodiment for implementing an instruction break scheme by a hardware mechanism. In the 17th embodiment, the 15th embodiment shown in FIG. **28** and the 16th embodiment shown in FIG. **34** are combined. In the 17th embodiment shown in FIG. **40**, the same reference numerals as in FIGS. **28** and **34** denote the same functional parts as in FIGS. **28** and **34**, respectively, and a detailed description thereof will be omitted.

The 17th embodiment shown in FIG. **40** is different from the 15th embodiment shown in FIG. **28** an the 16th embodiment shown in FIG. **34** in the construction of each of determination sections of an instruction break detection section **23**. Each of determination sections 170_0 to 170_n of this embodiment has a construction shown in FIG. **41**. FIG. **41** is a block diagram showing the construction of the determination section 170_0 as a representative of the determination sections 170_n to 170_n. In FIG. **40**, the same reference numerals as in FIGS. **29** and **35** denote the same blocks as in FIGS. **29** and **35**, respectively, and a detailed description thereof will be omitted.

As shown in FIG. **41**, the determination section 170_0 of this embodiment comprises a comparison section **101**, a conditional instruction decoder **102**, a condition determination section **103**, two AND circuits **151** and **152**, an OR circuit **153**, an unconditional instruction decoder **161**, and another OR circuit **162**.

The unconditional instruction decoder **161** decodes an instruction word that is supplied from an instruction register **22** and currently being executed to detect whether the instruction word is an unconditional instruction and supplies the decoding result to the OR circuit **162**. When the supplied instruction word is an unconditional instruction, the unconditional instruction decoder **161** output a signal having the value "1".

On the basis of the decoding result from the conditional instruction decoder **102**, which represents whether the instruction is a conditional instruction, the condition determination section **103** of this embodiment determines whether the condition designated by the condition code of the conditional instruction, which is supplied from a condition register **51**, is satisfied, and supplies the determination signal to the OR circuit **162**.

The OR circuit **162** performs OR operation to the condition determination signal obtained by the condition determination section **103** for the conditional instruction and the unconditional instruction determination signal obtained by the unconditional instruction decoder **161**, and outputs the OR operation result to one AND circuit **151**.

The AND circuit **151** performs AND operation to a signal output from the OR circuit **162**, a determination signal output from the comparison section **101** and related to an instruction break address, the value of a flag register **24***b* of a breakpoint register 24_0 provided in accordance with the determination section 170_0, and the value of a mode register **24***d*, and outputs the AND operation result to the OR circuit **153**.

The other AND circuit **152** performs AND operation to the determination signal output from the comparison section **101** and related to an instruction break address, the value of the flag register **24***b* of the breakpoint register 24_0 provided in accordance with the determination section 170_0, and a value obtained by inverting the value of the mode register **24***d*, and outputs the AND operation result to the OR circuit **153**. The OR circuit **153** performs OR operation to the signals output from the two AND circuits **151** and **152**, and outputs the OR operation result to an OR circuit **26** shown in FIG. **40**.

According to this construction, in a situation where the conditional instruction break mode is designated by mode information stored in the mode register **24***d*, and the instruction word stored in the instruction register **22** is a conditional

	Document ID	U	Title	Current OR
190	US 4399532 A	☒	Methods of and systems for monitoring a first call connection while effecting the establishment of a second call connection	370/262
191	US 4393269 A	☒	Method and apparatus incorporating a one-way sequence for transaction and identity verification	705/75
192	US 4284849 A	☒	Monitoring and signalling system	379/38
193	US 4273961 A	☒	Apparatus for communicating with processing apparatus over a telephone network	379/40
194	US 4259548 A	☒	Apparatus for monitoring and signalling system	379/38
195	US 4167781 A	☒	Microprocessor system having a single central processing unit shared by a plurality of subsystems each having a memory	717/127
196	US 4166289 A	☒	Storage controller for a digital signal processing system	710/33
197	US 4124891 A	☒	Memory access system	711/100
198	US 4117278 A	☒	Service observing terminal	379/32.01
199	US 4099234 A	☒	Input/output processing system utilizing locked processors	714/11
200	US 4084236 A	☐	Error detection and correction capability for a memory system	711/118

instruction, in at least one entry of the determination sections 170_0 to 170_n of the instruction break detection section **23**, when the instruction break address and current execution address match, the value of the flag register **24***b* is "1", and the condition of the conditional instruction is satisfied, the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

In a situation where the conditional instruction break mode is designated by mode information stored in the mode register **24***d*, and the instruction word stored in the instruction register **22** is an unconditional instruction, in at least one entry of the determination sections 170_0 to 170_n of the instruction break detection section **23**, when the instruction break address and current execution address match, and the value of the flag register **24***b* is "1", the OR circuit **26** outputs the interrupt notification signal **67** to the interrupt control section **40**.

In addition, in a situation where the instruction break mode is designated by mode information stored in the mode register **24***d*, in at least one entry of the determination sections 170_0 to 170_n of the instruction break detection section **23**, when the instruction break address and current execution address match, and the value of the flag register **24***b* is "1", the OR circuit **26** outputs the interrupt notification signal **67** to the interrupt control section **40**.

As described above, according to the 17th embodiment, when the conditional instruction break mode is designated, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. Additionally, when the instruction break mode is designated, a break-interrupt can be controlled independently of whether the condition of the conditional instruction is satisfied and in accordance with whether the instruction break generation condition is satisfied.

Furthermore, when the conditional instruction break mode is designated, and the supplied instruction word is a conditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition and the condition of the conditional instruction are satisfied. Also, even when the supplied instruction word is an unconditional instruction, a break-interrupt can be controlled in accordance with whether the instruction break generation condition is satisfied.

In the 17th embodiment, an construction in which the mode register **24***d* is added to the scalar processor described in the 10th embodiment, and the two AND circuit **151** and **152**, the OR circuit **153**, the unconditional instruction decoder **161**, and the OR circuit **162** are added in each of the determination sections 170_0 to 170_n has been described. These components may be added to the VLIW type processor and parallel processor described in the 11th to 14th embodiments.

In this case, the determination sections have constructions shown in FIGS. **42** to **45**, respectively. In FIGS. **42** to **45**, the same reference numerals as in FIGS. **21**, **23**, **25**, **27**, **35**, and **41** denote respectively the same parts as in FIGS. **21**, **23**, **25**, **27**, **35**, and **41**, which perform the same operations as described above. Each component with a symbol "'" has the same function as that of a corresponding component without this symbol, and its operation will be apparent without a detailed description of FIGS. **42** to **45**.

18th Embodiment

The 18th embodiment of the present invention will be described next with reference to drawings.

In the 10th to 17th embodiments, an example has been described in which whether the instruction break generation condition and the condition of a conditional instruction are satisfied is implemented by a hardware mechanism. In the 18th to 22nd embodiments to be described below, an example will be described in which whether at least the condition of a conditional instruction is satisfied is implemented by the function of software.

In the 18th to 21st embodiments, the overall construction of a data processing system (processor) for implementing the instruction break scheme is the same as that shown in FIG. **3**. As is apparent from this, whether the instruction break generation condition is satisfied is determined by a hardware mechanism using breakpoint registers 24_0 to 24_n for holding the target address of a breakpoint and flag information on whether the instruction break operation is valid, and determination sections 25_n to 25_n for determining on the basis of these pieces of information whether the instruction break generation condition is satisfied.

In this case, each of the determination sections 25_0 to 25_n has a construction shown in FIG. **46**. FIG. **46** is a block diagram showing the construction of the determination section 25_0 as a representative of the determination sections 25_0 to 25_n In FIG. **46**, the same reference numerals as in FIG. **19** denote the same blocks as in FIG. **19**, respectively.

As shown in FIG. **46**, the determination section 25_0 of this embodiment comprises a comparison section **101** and AND circuit **104**.

The comparison section **101** compares an instruction break address held in an address register **24***a* of the breakpoint register 24_0 provided in accordance with the determination section 25_0 with a current execution address supplied from a program counter **21** and determines whether the two addresses match. When the two addresses match, the comparison section **101** outputs a determination signal having the value "1". When the two addresses do not match, the comparison section **101** outputs a determination signal having the value "0".

The AND circuit **104** performs AND operation to the value of a flag register **24***b* of the breakpoint register 24_0 provided in accordance with the determination section 25_0 and the determination signal output from the comparison section **101** and related to the instruction break address and outputs the AND operation result to an OR circuit **26** shown in FIG. **3**.

According to this construction, in at least one entry of the determination sections 25_0 to 25_n of an instruction break detection section **23**, when the instruction break address and current execution address match, and the value of the flag register **24***b* is "1", the OR circuit **26** outputs an interrupt notification signal **67** to an interrupt control section **40**.

The interrupt notification signal **67** output in this embodiment represents only that the instruction break generation condition is satisfied and does not includes that the condition of the conditional instruction is satisfied. Whether the condition of the conditional instruction is satisfied is determined by an interrupt handler as a program for condition determination, which is stored in a memory **10** and started as the interrupt notification signal **67** is received.

More specifically, when receiving the interrupt notification signal **67** from the OR circuit **26**, the interrupt control section **40** reads out an instruction address **73** at the time of